DESCRIPTION AND IMPLEMENTATION OF NUMBER THEORETIC TRANSFORMS

Antonio Catarino Rodrigues de Sousa

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESCRIPTION AND IMPLEMENTATION OF
NUMBER THEORETIC TRANSFORMS

by

Antonio Catarino Rodrigues de Sousa

December 1978

Thesis Advisor: S. R. Parker

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Description and Implementation of Number Theoretic Transforms

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis; December 1978

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Antonio Catarino Rodrigues de Sousa

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE
December 1978

13. NUMBER OF PAGES
206

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Number Theoretic Transform
Fourier Transform
Fermat Number Transform

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis summarizes the theory of number theoretic transforms (NTT's), and presents original examples to illustrate the theory. Concepts have been studied and compared in order to present them in a cohesive and unified manner.

Software and hardware implementation of Fermat number transforms are discussed and compared with the Fourier

(20. ABSTRACT Continued)

Transform showing a substantial improvement in efficiency and accuracy. The main drawback of Fermat Number Transforms is a rigid relationship between the allowed sequence length and word length. Methods and other NTT's, for overcoming this problem are discussed. The theory has also been extended to two dimensions.

Approved for public release; distribution unlimited.

Description and Implementation of Number Theoretic Transforms

by

Antonio Catarino Rodrigues de Sousa
Lieutenant, Portuguese Navy
B.S.E.E., Naval Postgraduate School, 1977

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

December 1978

ABSTRACT

This thesis summarizes the theory of number theoretic transforms (NTT's), and presents original examples to illustrate the theory. Concepts have been studied and compared in order to present them in a cohesive and unified manner.

Software and hardware implementation of Fermat number transforms are discussed and compared with the Fourier Transform showing a substantial improvement in efficiency and accuracy. The main drawback of Fermat Number Transforms is a rigid relationship between the allowed sequence length and word length. Methods and other NTT's, for overcoming this problem are discussed. The theory has also been extended to two dimensions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF DRAWINGS

ACKNOWLEDGEMENT

The author expresses his sincere appreciation to Professor S. R. Parker for his guidance in this study. He also wishes to thank his wife Isabel, whose moral support is present in every page of this thesis.

I. INTRODUCTION

With the rapid advances in large scale integration, a growing number of complex digital signal processing applications are becoming economically feasible. In most cases the bulk of processing workload appears to consist of digital filter computation. Future progress in digital signal processing, either towards high speed, real time operation or increased sophistication, thus largely depends on increased efficiency in digital filtering computation. This can be achieved not only by implementing improved filter circuits but also by using better computational algorithms as will be discussed in this thesis.

Schonhaje and Strussen [1] (also see text by Knuth [2,p.269]) defined Fourier-like transforms over the ring of integers, modulo the Fermat numbers [2] $2^{2^n} + 1$, to yield convolutions. They showed that such convolutions can be used to perform fast integer multiplications. Rader [3] and Agarwal and Burrus [4] also defined Fourier-like transforms over residue classes of integers, modulo the Fermat and Mersenne primes, to compute convolutions of the real integer sequences.

For this presentation and exposition of Number Theoretic Transforms (NTT), the works of many different authors were consulted. Similar concepts have been studied and compared in order to present them in a cohesive and unified manner.

In section II a mathematical framework for these new transforms is presented. This framework is based on the theory of congruences, modulo M, which belongs to the general area of what is often called "number theory." Number theory is very old, going back several thousand years; at least as far back as Euclid, who proved some of the original results. The names of many other famous mathematicians are also associated with the theory, including Joseph Lagrange (1763-1813), and Leonard Euler (1707-1783), and Carl Gauss (1777-1855) who is responsible for many contributions to the area, some of which were published in his book Disquisitiones Arithmeticae, published in 1801, when he was 24 [5]. An excellent history of the theory of numbers is found in the work of Dickson [6].

In recent years, there has been increasing interest in the practical applications of various parts of number theory, including the theory of residue number systems. There has been some work on the use of these number systems in general-purpose computers [7]-[12], although this line of investigation has not yielded many practical results due to the difficulty of determining the sign of numbers expressed in residue number system notation. More promising results have been obtained in applications where sign detection is not required, such as number theoretic transforms [3], [4] and [13].

The possibility of performing the fast Fourier transform (FFT) in finite algebraic systems [14], [15] and [1], is

being increasingly discussed as a means of digital filtering [3], [13] and [16] (see also references in Reference [13]). The convolution property which such transforms share with the conventional f.f.t. can be employed to construct a non-recursive digital filter in which rounding error does not occur.

The following type of transforms have been discussed in the literature.

(a) Transforms in arithmetic mod p, p prime, in which the order (number of points), d, for example, is a factor of p-1.

(b) Transforms in arithmetic mod m, m arbitrary, in which d dividies p-1 for each prime factor p of m.

(c) Transforms in an arbitrary finite Galois field $GF(p^n)$ of p^n elements, where d divides p^n-1.

Here class (a) is treated as a special case of both (b) and (c), which are essentially different. This material is presented in section III which discusses the Fourier transform in a finite field in a broad way.

The best known number-theoretic transform is the Fermat-number transform [1], [3], [4], [13], [16]. Due to the simplicity of its arithmetic, such a transform is the fastest method known so far for computing integer convolutions under certain conditions. However, this transform suffers from the disadvantage that the restriction imposed on the

register word length is often too excessive [13]. In order to remedy this problem, Rader [3] has suggested using a two-dimensional convolution scheme to convolve long one-dimensional sequences, and Agarwal and Burrus [17], [18] have presented such a two-dimensional convolution scheme. Other authors [19], [20], [21], [22], [23] have also considered other number-theoretic transforms (NTT). Section IV discusses the various types of NTT.

These transforms provide more choices of word length and transform length, at speeds which are not attainable by the Fermat-number transforms under similar conditions. Problems involved in the implementation of Fermat-number transforms are discussed in section V.

Section VI deals with the comparison between the FFT and the FNT. Software and hardware requirements for both are analyzed and compared. Agarwal [13] programmed Fermat number transforms on the IBM 370/155 computer [13] and showed how to compute convolutions approximately three times as fast as the FFT implementation for the same convolution. However, their main drawback is a rigid relationship between word length and obtainable transform length, as well as a limited choice of possible word lengths. This last point is especially significant for FNT's, and may result in a waste of computing power when the permissible word lengths do not correspond to the dynamic range required for the convolution.

In principle, Number Theoretic Transforms (NTT) could be implemented in the same was as Discrete Fourier Transforms with multiplications by trigonometric functions replaced by multiplications by powers of two, all operations being performed modulo a Mersenne or Fermat Number. When the transforms have a composite number of terms, as is the case with Fermat Number Transforms (FNT) or some pseudo-Mersenne Transforms [24], various pipeline computing techniques can be used [25].

In practice, however, direct transposition of Fast Fourier Transforms (FFT) architectures does not necessarily lead to optimum implementations and the development of special configurations for computing NTT seems worth exploring. Along these lines, McClellan [26] has proposed a new coding technique for simplifying the implementation of Fermat Number Transforms.

McClellan implemented a FNT convolver for radar signal processing and reached some interesting conclusions. Leibowitz [27] presents a code translation which is mathematically simpler, and this proposed arithmetic provides simpler realizations of all operations required to compute the FNT.

Nussbaumer [28] discusses the implementation of pseudo-Mersenne and Fermat Number Transforms. He shows that some pseudo-Mersenne Transforms can be computed efficiently by a linear filtering approach. This approach is extended to

cover the case of Fermat and pseudo-Fermat Number Transforms by using a special coding scheme for implementing arithmetic operations in a Fermat Number system. A number of suggestions have arisen for lengthening the sequences which can be handled by the NTT. One suggestion is to perform the calculation modulo several mutually prime modulo, and then obtain the desired result by using the Chinese Remainder theorem [13], [29]. Reed and Truong [30] have also shown how one can extend the method to Galois fields over complex integers modulo Mersenne primes to enable one to use the FFT algorithm to compute convolutions of complex sequences, and to lengthen the sequences which the method can handle. However, because this method requires several multiplications, it does not seem very promising.

One of the most promising methods for lengthening the sequence one can handle has been suggested by Rader [3] and developed by Agarwal and Burrus [31]. This consisted of mapping the one-dimensional sequences into multidimensional sequences and expressing the convolution as a multidimensional convolution. In Appendix A an explanation of the process of two dimensional convolution for convolving long sequences is presented. By the use of an example it is shown that this process improves the length of the sequences handled by the NTT.

With knowledge of the advantages and disadvantages of Fermat Number Transforms, it is possible to speculate on just what type of problems are likely to benefit from this

new approach. In general one looks for problems which have the following characteristics:

1) Fairly short sequences (about fifty delayed products)
2) A high accuracy requirement
3) Implementations where multiplications are very much more costly than additions.

Two specific situations come to mind. The first is the estimation of magnitude spectra of (many simultaneous) wideband signals. The theory of power spectra estimation states that power density computations involves formulating a correlation function with a finite number of delays which are a fraction of the number of data points available.

The second situation is two dimensional finite impulse response filtering. Here we may consider an arbitrary L x L impulse response to be applied to a large image. If L is in the range of 52 points, the FFT is not particularly attractive for convolution, although more so for two dimensional convolution than for one dimensional convolution.

The Fermat Number Transform is quite effective, however. For impulse responses of this size, the number of multiplications is reduced by about two orders of magnitude over the direct method, in exchange for a number of additions which are not too different (usually less) than required for the direct method. Thus it can be expected that the Fermat number transform will soon play a part in the computation required for the filtering of pictures.

Recently, Derome [32] discusses a class of NTT's based on three bit primes, having many of the computational advantages of the FNT's. These NTT's can have much larger transform lengths than those for FNT's so that the fast convolution of, for example, a 1000 x 1000 point picture with a 24 x 24 point spread function should be possible in a minicomputer! This development was undertaken in connection with analysis of high resolution electron micrographs.

II. MODULAR ARITHMETIC

Let a and b be integers. We say that "a divides b" (denoted by "$a|b$") and "b is multiple of a" if there exists an integer c such that $b = ac$.

If a does not divide b, we denote the fact by "$a \nmid b$". An important theorem concerning the division of integers (the Division Algorithm), is:

Let a and b be integers, b not zero. Then there exist two unique integers, q and r, such that

$$a = bq + r \quad \text{and} \quad 0 \leq r \leq b \tag{2.1}$$

The integers q and r, are called the "quotient" and the "remainder" respectively.

If a, b and M, are integers, with $M > 0$, such that

$$M|(a - b) \tag{2.2}$$

we say that "a is congruent to b, modulus M", and we denote this fact, by writing

$$a \equiv b \pmod{M} \tag{2.3}$$

In other words, two integers a and b are congruent mod M, if M divides their difference.

We will refer to M as the "modulus". Note that

1) $23 \equiv 8 \pmod 5$ since $23 - 8 = 15 = 5 \cdot 3$

and

2) $23 \equiv 3 \pmod 5$ since $23 - 3 = 20 = 5 \cdot 4$

are both true statements.

We restrict our attention to what is called a <u>complete residue system, mod M</u>, as the set of integers

$$Z_M = \{0, 1, 2, \ldots, M-1\}. \qquad (2.4)$$

In other words, when an integer a is divided by another M i.e.,

$$a = KM + b$$

where the remainder b, is some positive integer less than M, there exists a congruence

$$a \equiv b \pmod M \qquad (2.5)$$

such that b is a unique integer among the numbers

$$0, 1, 2, \ldots M-1 .$$

Thus, one possible mechanization of residue reduction, is to divide by the modulus and keep only the remainder. Examples:

1) $47 \equiv 2 \pmod 9$ since $\frac{47}{9} = 5$ remainder = 2

2) $81 \equiv 0 \pmod{27}$ since $\frac{81}{27} = 3$ remainder = 0

Both 2 and 0 are contained only once within the set of integers {0,1,...,8} and {0,1,2, ..., 26} respectively. The last example shows that, in general instead of saying that a number a is divisible by the number M we can write

$$a \equiv 0 \pmod M$$

For this means $a - 0 = a = Mk$, where k is some integer.

For instance, instead of saying that a is an even number, we can write

$$a \equiv 0 \pmod 2$$

In the same manner one sees that an odd number satisfies

$$a \equiv 1 \pmod 2.$$

In working with residue reduction we will drop the symbol $\equiv$ for congruence and will use the symbol =.

But congruence is not the same as equality unless one can show separately that the difference between a and b is also less than M. The following basic arithmetic operations are permissible with modular arithmetic:

a) Addition: $2 + 6 = 8 = 1 \pmod 7$

b) Negation: $-2 = -2 + 7 = 5 \pmod 7$

c) Subtraction: $3 - 5 = 3 + (-5) = 3 + (-5 + 7) = 5 \pmod 7$

d) Multiplication: $3 \times 6 = 18 = 4 \pmod 7$

e) Multiplicative inverse: Multiplicative inverse of an integer b in Z_M exists if and only if b and $\underline{M}$ are relative primes. In that case $b \times b^{-1} = 1 \pmod M$ $2^{-1} = 4 \pmod 7$; $2 \times 4 = 8 = 1 \pmod 7$.

f) Division: a/b exists if and only if b has an inverse. In that case $a/b = a \times b^{-1}$; $4/2 = 4 \times 4 = 16 = 2 \pmod 7$.

Note that because of the nature of modular arithmetic, numbers do not have sizes or magnitude. One cannot say that a particular number is larger than another or that numbers are close.

Extracting the residue is a functional transformation of a into b. It occurs often enough in what follows that it deserves a special symbol, $\langle \cdot \rangle$.

$$\langle a \rangle_M = b \qquad (2.6)$$

The subscript M may be omitted if it is understood from the context.

Computations involving residues are usually simple because it is never necessary to work with quantities larger than the modulus unless one finds it convenient.

Notice that

$$\begin{aligned} &\langle x+y \rangle \text{ is the same as } \langle\langle x \rangle + \langle y \rangle\rangle \\ &\langle x-y \rangle \text{ is the same as } \langle\langle x \rangle - \langle y \rangle\rangle \qquad (2.7) \\ &\langle xy \rangle \text{ is the same as } \langle\langle x \rangle \langle y \rangle\rangle \end{aligned}$$

so that in any computation involving only +, -, x one may, at their option, replace the result of any step by its residue. Example:

$$\langle 15+13 \rangle_7 = \langle\langle 15 \rangle_7 + \langle 13 \rangle_7 \rangle_7 = \langle 1+6 \rangle_7 = 0$$

$$\langle 12-11 \rangle_7 = \langle\langle 12 \rangle_7 - \langle 11 \rangle_7 \rangle = \langle 5 - 4 \rangle_7 = 1$$

$$\langle 9 \times 14 \rangle_7 = \langle\langle 9 \rangle_7 \times \langle 14 \rangle_7 \rangle = \langle 2 \times 0 \rangle_7 = 0$$

Note that if it were necessary to divide for residue reduction the operation would be quite costly.

Fortunately there are simpler techniques. In the simplest case - the residue or a decimal number modulo 10 is its last decimal digit, since

$$\langle a \rangle_{10} = \langle \sum_i a_i \, 10^i \rangle_{10} = \langle \sum_i \langle a_i \, 10^i \rangle \rangle \qquad (2.8)$$

and $\langle a_i \, 10^i \rangle_{10}$ is zero except for i = 0 term.
Example:

$$\langle 1098 \rangle_{10} = \langle \sum_{i=0}^{3} a_i \, 10^i \rangle_{10}$$

$$= \langle 1 \times 10^3 + 0 \times 10^2 + 9 \times 10^1 + 8 \times 10^0 \rangle_{10}$$

$$\langle 1098 \rangle_{10} = 8 \pmod{10}$$

This can be generalized to any radix. If M is a power of two, and a is represented on a binary machine, one has a trivial method of extracting $\langle a \rangle_M$.

$$\langle a \rangle_{2^K} = \langle \sum_i a_i \langle 2^i \rangle \rangle = \sum_{i=0}^{K-1} a_i \, 2^i \qquad (2.9)$$

This operation is performed by "masking out" all but the K least significant bits.

Example:

$$\langle 15 \rangle_{2^3} = \langle \sum_{i=0}^{2} a_i \langle 2^i \rangle \rangle = \sum_{i=0}^{2} a_i \, 2^i$$

$$= \langle 1111 \rangle_{2^3} = 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 = 7 \bmod 8$$

i.e., here K = 3, and only the 3 least significant bits account for the value of the residue. The fourth bit (the most significant bit in this case) is "masked out."

A. SOME IMPORTANT RESULTS IN MODULAR ARITHMETIC

Euler's function is defined as $\phi(M)$, the number of integers in the finite set {0,1,2, ... M-1} (which is called the set of integers mod M, and denoted by Z_M) that are relatively prime to M.

For M a prime,

$$\phi(M) = M - 1 \qquad (2.10)$$

Example: let M = 7. The ring of integers mod 7 is $Z_7 = \{0,1,2,3,4,5,6\}$. Since M = 7, is a prime, the integers in Z_7 that are relatively prime to M = 7, are all elements of the set (except zero), i.e.

$$\phi(M) = M - 1$$

$$\phi(7) = 7 - 1 = 6$$

If M is composite and its prime factored form is denoted by

$$M = p_1^{r_1} p_2^{r_2} \ldots p_\ell^{r_\ell} \tag{2.11}$$

then the general expression for $\phi(M)$ is, [13]

$$\phi(M) = M\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \ldots \left(1 - \frac{1}{p_\ell}\right) \tag{2.12}$$

Example: Let $M = 12 = 2^2 \times 3$

$$Z_{12} = \{0,1,2,3,4,5,6,7,8,9,10,11\}$$

by simple counting one finds 4 numbers that satisfy the conditions of being relatively prime to M. Checking by applying equation (2.12)

$$\phi(12) = 12\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 12\left(\frac{1}{2}\right)\left(\frac{2}{3}\right) = 4.$$

An important theorem known as Euler's theorem states that for every α relative prime to M

$$\alpha^{\phi(M)} = 1 \pmod{M} \tag{2.13}$$

For M prime, this reduces to Fermat's theorem

$$\alpha^{M-1} = 1 \pmod{M} \tag{2.14}$$

which holds for all nonzero elements of Z_M, since they are all relative prime to M, if M is prime by assumption.

Example: Let M = 7

$$Z_7 = \{0,1,2,3,4,5,6\}$$

$$\phi(7) = 7 - 1 = 6$$

$$\alpha^{\phi(M)} = \alpha^{M-1} = 1 \pmod{M}$$

which holds for all non-zero elements of Z_M since they are all relative primes to M.

Here:

$$1^6 = 1 \pmod 7$$

$$2^6 = 64 = 1 \pmod 7$$

$$3^6 = 729 = 1 \pmod 7$$

$$4^6 = 4096 = 1 \pmod 7$$

$$5^6 = 15625 = 1 \pmod 7$$

$$6^6 = 46656 = 1 \pmod 7$$

There are certain roots of unity that are of particular

interest. If N is the least positive integer such that

$$\alpha^N = 1 \pmod{M} \tag{2.15}$$

then α is said to be a root of unity of order M, or simply of order N.

If the order of α is equal to $\phi(M)$, then α is called a primitive root.

If M is prime and α is a primitive root the set of integers

$$\{\alpha^K \pmod{M}, \quad K = 0,1,2, \ldots M-2\} \tag{2.16}$$

is the total set of non-zero elements in Z_M. Thus all nonzero integers in Z_M can be generated by powers of a primitive root. This characterizes the entire field.

Example:

Let M = 7

$$Z_7 = \{0,1,2,3,4,5,6\}$$

Looking in a table of primitive roots, for example [13] we get

3 and 5 are primitive roots of 7,

thus

$$\{3^K \pmod 7,\ K = 0,1,2,\ldots,5 = M-2\}$$

$$= \{3^0,3^1,3^2,3^3,3^4,3^5\}$$

$$= \{1,3,2,6,4,5\} \bmod 7$$

and those are all non-zero elements in Z_7. The same for

$$\{5^K \pmod 7,\ K = 0,1,2,\ldots,5\}$$

$$= \{5^0,5^1,5^2,5^3,5^4,5^5\}$$

$$= \{1,5,4,6,2,3\} \bmod 7.$$

Euler's theorem implies that if α is of order N, then N must divide $\phi(M)$, denoted by

$$N \mid \phi(M) \tag{2.17}$$

If M is a prime it can be shown [13], that roots of order N exist if and only if

$$N \mid (M-1) \tag{2.18}$$

and the roots are given by [13],

$$\alpha = \alpha_\phi^{M-1/N} \tag{2.19}$$

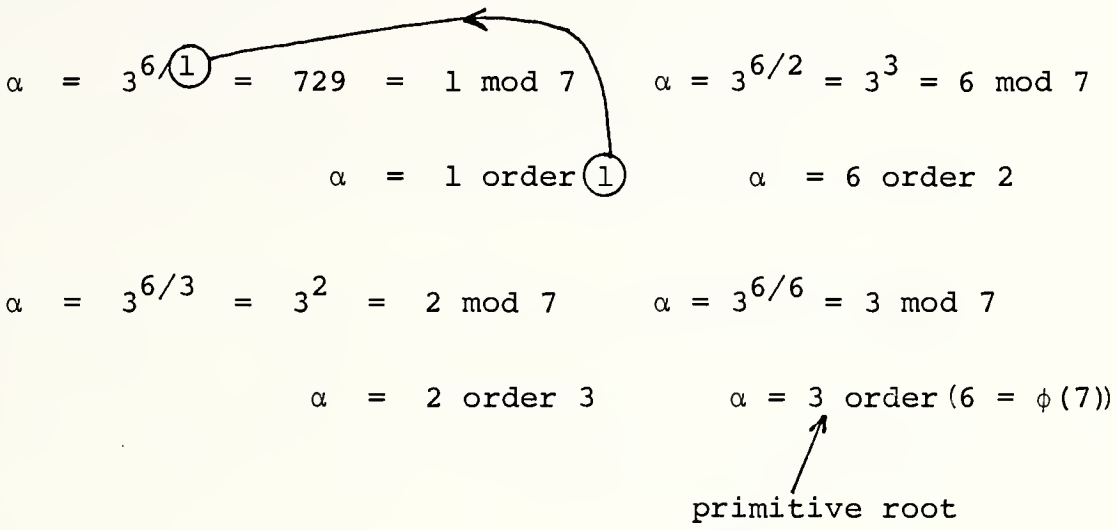
where α_ϕ denotes a primite root.

Example: Let M = 7 $\phi(7) = M - 1 = 6$

Possible order of roots: N = 1,2,3,6

primitive root (from table): 3

Order of roots:



More generally, if α is a root of order N then [13]

α^K is of order N/K, if $K|N$

(2.20)

α^K is of order N, if N and K are relatively primes

Example: Let M = 11 $\phi(M) = 10$

Now $\alpha = 2$ is a primitive root since $2^{10} = 1 \bmod 11$ and N = 10 is the least positive number such that $\alpha^N = 2^{N=10} = 1 \bmod 11$

Then

$$\alpha^K = 2^2 = 4 \bmod 1 \quad \text{is a root of order } {}^{N=10}/_{K=2} = 5$$

This can be seen as follows:

$N =$	0	1	2	3	4	5	6	7	8	9	10
4^N	1	4	5	9	3	1	4	5	9	3	1

i.e., the root $\alpha = 4$ generates a cyclic subset of the field with N = 5 distinct elements (order 5).

For $\alpha^K = 2^3 = 8 \bmod 11$ since N = 10 and K = 3 are relatively primes $\alpha = 8 \bmod 11$ will be a root of order N = 10, or a primitive root.

Checking:

$N =$	0	1	2	3	4	5	6	7	8	9	10
8^N	1	8	9	6	4	10	3	2	5	7	1

Notice that (2.20) implies that the number of roots of order N, is given by $\phi(M)$, and therefore the number of primitive roots is $\phi(\phi(M))$ (since for a primitive root $N = \phi(M)$).

Example: Let M = 7 $\quad Z_7 = \{0,1,2,3,4,5,6\}$

Number of primitive roots = $\phi(\phi(7)) = \phi(6) = 6(1-\frac{1}{2})(1-\frac{1}{3}) = 2$

as seen previously, 3 and 5 are primitive root mod 7.

Number of roots of order N = 3: $\phi(3) = 3 - 1 = 2$.

$\alpha = 3^{6/3} = 3^2 = 2 \bmod 7$ $\qquad$ $\alpha = 5^{6/3} = 5^2 = 4 \bmod 7$

$\alpha = 2$ order 3 $\qquad$ $\alpha = 4$ order 3

So $\alpha = 2 \bmod 7$ and $\alpha = 4 \bmod 7$ roots of order N = 3. These relations will allow one to calculate all of the roots of all possible orders from one primitive root.

We can summarize these ideas more precisely in the following:

- The highest possible exponent to which a number can belong mod M is $\phi(M)$
  (i.e. the highest order of a root is $N = \phi(M)$)
- If a number has order $N = \phi(M)$, we shall call it a primitive root for the modulus M.
- Not every modul has primitive roots; for instance, for the modulus M = 15, one finds that every x in Z_M, relatively prime to 15 satisfy the congruence

$$x^4 \equiv 1 \pmod{15}$$

  and yet $\phi(15) = 8$.
- To find the primitive roots of a modul if they exist, one must usually proceed by trial and error, although there are certain rules that may facilitate the search.

  Often one of the small number 2, 3, 5 or 6 may turn out to be a primitive root.

- Extensive tables of primitive roots for primes have been computed. The first of these, the "Canon Arithmeticus" (1839) by K.G.J. Jacobi, included primitive roots for all primes below 1,000. More recent tables by Kraitchick, Cunningham, and others give primitive roots for all primes up to 25,000 and even beyond.

One interesting point, not mentioned so far, is the existence of multiplicative inverse.

Multiplicative inverse of an integer b in Z_M exists if and only if b and M are relatively primes.

In that case, b^{-1} is one integer such that

$$b \times b^{-1} = 1 \pmod{M} \tag{2.21}$$

Example: Let M = 7

$5^{-1} = ?$ $5 \times 5^{-1} = 1 \bmod 7$ $5^{-1} = 3$

Since $5 \times 3 = 15 = 1 \bmod 7$, i.e., if M is a prime for every non-zero integer α, in Z_M, there exists an inverse [13]

$$\alpha^{M-2} \tag{2.21a}$$

this can be seen by another example.

Example: $M = 7$

$1^{-1} = 1^{7-2} = 1^5 = 1 \bmod 7 \qquad 1 \times 1^{-1} = 1 \bmod 7$

$2^{-1} = 2^{7-2} = 2^5 = 32 = 4 \bmod 7 \qquad 2 \times 2^{-1} = 2 \times 4 = 8 = 1 \bmod 7$

$3^{-1} = 3^{7-2} = 3^5 = 243 = 5 \bmod 7 \qquad 3 \times 3^{-1} = 3 \times 5 = 15 = 1 \bmod 7$

$4^{-1} = 4^{7-2} = 4^5 = 1024 = 2 \bmod 7 \qquad 4 \times 4^{-1} = 4 \times 2 = 8 = 1 \bmod 7$

$5^{-1} = 5^{7-2} = 5^5 = 3125 = 3 \bmod 7 \qquad 5 \times 5^{-1} = 5 \times 3 = 15 = 1 \bmod 7$

$6^{-1} = 6^{7-2} = 6^5 = 7776 = 6 \bmod 7 \qquad 6 \times 6^{-1} = 6 \times 6 = 36 = 1 \bmod 7$

Note that for a non-prime M, α has an inverse given by [13]:

$$\alpha^{\phi(M)-1} \tag{2.22}$$

if α and M are relatively primes.

Example: $N = 12 = 2^2 \times 3$

$$\phi(12) = 12(1 - \tfrac{1}{2})\ (1 - \tfrac{1}{3}) = 12(\tfrac{1}{2})\ (\tfrac{2}{3}) = 4$$

$$Z_{M=12} = \{0,1,2,3,4,5,6,7,8,9,10,11\}$$

So there are $\phi(12) = 4$ elements in Z_{12}, (1,5,7,11) that are relatively prime to M = 12.

Those α's have inverses given by:

$\alpha_1 = 1 \quad 1^{-1} = 1^{\phi(12)-1} = 1^{4-1} = 1^3 = 1 \bmod 12$

$1 \times 1^{-1} = 1 \times 1 = 1 \bmod 12 \quad = 1 \bmod 12$

$\alpha_2 = 5 \quad 5^{-1} = 5^{\phi(12)-1} = 5^{4-1} = 5^3 = 5 \bmod 12$

$5 \times 5^{-1} = 5 \times 5 = 1 \bmod 12$

$\alpha_3 = 7 \quad 7^{-1} = 7^{\phi(12)-1} = 7^{4-1} = 7^3 = 7 \bmod 12$

$7 \times 7^{-1} = 7 \times 7 = 1 \bmod 12$

$\alpha_4 = 11 \quad 11^{-1} = 11^{\phi(12)-1} = 11^{4-1} = 11^3 = 11 \bmod 12$

$11 \times 11^{-1} = 11 \times 11 = 1 \bmod 12$

This suggests that in this case $b^{-1} = b$. Therefore, we verify that by considering M a composite rather than a prime, one observes several differences.

If we define Z_M as the ring of integers modulo M, then if in a ring of integers multiplicative inverses exist for all non-zero integers, this ring is called a field.

A field with a finite number of elements is also called a Galois field. It is clear then, that Z_M is a field if and only if M is a prime.

Z_M is not a field for a non prime M, since not all elements will have multiplicative inverses. Also, there is no primitive root that will generate the entire ring, (only subsets with $\phi(M)$ elements, at most.)

Example: $M = 8 = 2^3$

$$\phi(M) = 8(1 - \frac{1}{2}) = 4$$

N	0	1	2	3	4	5	6	7	
1^N	1	1	1	1	1	1	1	1	
2^N	1	2	4	0	0	0	0	0	subset with 4 elements
3^N	1	3	1	3	1	3	1	3	root of order N = 2
4^N	1	4	0	0	0	0	0	0	subset with 3 elements
5^N	1	5	1	5	1	5	1	5	root of order N = 2
6^N	1	6	4	0	0	0	0	0	subset with 4 elements
7^N	1	7	1	7	1	7	1	7	root of order 2

In these conditions, the previous example shows, that there are multiplicative inverses, only for those elements in Z_8 relative primes to M = 8.

We can investigate, a little more, the arithmetic modulo M, when M is not a prime. Let M have the following unique prime power factorization (2.11)

$$M = p_1^{r_1} \cdot p_2^{r_2} \cdots p_\ell^{r_\ell}$$

when the arithmetic is done mod M, it is in effect done modulo each prime power $p_i^{r_i}$ simultaneously [13].

A set of arithmetic operations can be done either modulo each $p_i^{r_i}$ separately and the final result mod M obtained using the Chinese remainder theorem [13], or

alternatively all the operations may be done mod M, but they must be valid operations mod for each $p_i^{r_i}$.

In these conditions, an integer α is said to be of order N in Z_M if and only if it is of order N in each $Z_{p_i^{r_i}}$.

Here we present some basic results.

$$a = b \bmod M$$

is true if and only if

$$a = b \bmod p_i^{r_i} \qquad i = 1,2,\ldots\ell \qquad (2.23)$$

If we know the residues of an integer a modulo each $p_i^{r_i}$. We can uniquely reconstruct the integer a mod M using the Chinese Remainder Theorem.

To establish this theorem, let

$$a = a_i \bmod p_i^{r_i} \qquad (2.24)$$

$$d_i \triangleq M/(p_i^{r_i}) \qquad (2.25)$$

and

$$d_i^{-1} \triangleq (d_i \bmod p_i^{r_i})^{-1} \bmod p_i^{r_i} \qquad (2.26)$$

then

$$a = \left(\sum_{i=1}^{p} d_i \, d_i^{-1} \, a_i \right) \bmod M \qquad (2.27)$$

Example:

Let $a = 123$ and $M = 24 = 2^3 \times 3$

Calculation of a_i,s:

$$a = 123 = a_1 \bmod 2^3 \qquad a_1 = 3 \bmod 8$$

$$a = 123 = a_2 \bmod 3 \qquad a_2 = 0 \bmod 3$$

Calculation of d_i,s:

$$d_1 = M/p_1^{r_1} = 24/2^3 = 3 \bmod 8$$

$$d_2 = M/p_2^{r_2} = 24/3 = 8 = 2 \bmod 3$$

Calculation of d_i^{-1}'s:

$$d_1^{-1} \triangleq (d_1 \bmod p_1^{r_1})^{-1} = (3 \bmod 8)^{-1} = 3^{\phi(8)-1} = 3^3 = 3 \bmod 8$$

$$d_2^{-1} \triangleq (d_2 \bmod p_2^{r_2})^{-1} = (2 \bmod 3)^{-1} = 2^{M02} = 2^{3-2} = 2 \bmod 3$$

then

$$a = \left(\sum_{i=1}^{2} d_i \, d_i^{-1} \, a_i \right) \bmod M$$

$$a = (3 \times 3 \times 3) + (2 \times 2 \times 0) = 27 = 3 \bmod 24$$

Check:

$$a = 123 = 3 \bmod 24.$$

III. TRANSFORMS IN FINITE FIELDS

The basic operations of signal filtering is convolution. In the discrete time signal processing situation, convolution takes the form

$$y(n) = x(n) * h(n) = \sum_{n=-\infty}^{\infty} x(m)\, h(n-m) \qquad (3.1)$$

$$n = 0, 1, 2, \ldots$$

Where $h(n)$ is the response to a unit impulse, for a causal filter, then

$$h(n) = 0 \quad \text{for} \quad n < 0 .$$

and, when the duration of the impulse response is finite, the infinite sum (3.1) reduces to a finite sum

$$y(n) = \sum_{m=0}^{N-1} x(m)\, h(n-m) \qquad (3.2)$$

where N is the length of the finite impulse response (FIR) filter.

The processing workload required to evaluate (3.2) can be reduced significantly if direct computation is replaced by transforms methods. This is so provided the

application in question allows sequences to be processed in blocks.

The Discrete-Fourier Transform (DFT) is one of the most versatile transforms, and is defined by

$$\text{DFT} \triangleq X(K) = \sum_{n=0}^{N-1} x(n)\, e^{-j(\frac{2\pi}{N})nK} \qquad (3.3)$$

$$K = 0, 1, \ldots N-1$$

and its inverse transform by

$$\text{IDFT} \triangleq x(n) = \frac{1}{N} \sum_{K=0}^{N-1} X(K)\, e^{j(\frac{2\pi}{N})nK} \qquad (3.4)$$

$$m = 0, 1, \ldots N-1$$

where N is the length of the sequence which transforms one wants to calculate.

In order to use the DFT in the high-speed implementation of convolution, one makes use of its cyclic convolution property, which states that

$$\text{DFT}[h(n) * x(n)] = \text{DFT}[h(n)] \cdot \text{DFT}[x(n)] \qquad (3.5)$$

and this implies that convolution can be implemented using

$$y(n) = IDFT\{DFT[h(n)] \cdot DFT[x(n)]\} \qquad (3.6)$$

The convolution implemented by (3.6) is called cyclic convolution since it evaluates (3.2) as if h(n) and x(n) were periodically extended outside the range [0,N-1], or equivalently, the indices were evaluated modulo N. Notice that normal finite convolution can be calculated by cyclic convolution if zeros are apended to x(n) and h(n) to prevent folding or aliasing [34].

The DFT is a transform of finite sequences, of real or complex numbers. One might ask if there are transforms in other number fields. That is, given a sequence of numbers modulo M, is there a transform that has the cyclic convolution property? One will see that the answer to this question is yes.

If one has a sequence of numbers of length N, then a transform of the form given by the pair

$$X(K) = \sum_{n=0}^{N-1} x(n)\, \alpha^{nK} \qquad (3.7)$$

$$x(n) = \frac{1}{N} \sum_{K=0}^{N-1} X(K)\, \alpha^{-nK} \qquad (3.7a)$$

is said to have a DFT structure [34].

If one looks for the properties that a general transform (3.1) having the DFT structure must have to exhibit the cyclic convolution property, one finds [4], [35], that it depends on the existence of an α that is a root of unity of order N, i.e.,

$$\alpha^N = 1 \tag{3.8}$$

and that $N^{-1} = \frac{1}{N}$ exists, i.e., the inverse of N exists. It has been shown [35] that in the complex number field the conventional DFT with $\alpha = e^{-j\frac{2\pi}{N}}$ is the only transform, with that property. However, by working in a finite field or ring of integers with arithmetic carried out an integer M, a large class of transforms exist [14] that have the cyclic convolution property. By special choices of the length N, the modulo M, and the value α, it is possible to have transforms with many interesting properties. These are called number theoretic transforms.

The possibilities of interest are, [14]:

1) the ring of integers Z_M, with respect to modulo M
2) the field of integers GF(p), with respect to prime modulus p
3) the Galois field $GF(p^k)$ of p^k elements.

Let Z_M represent the ring of integers {0,1 ... M-1} (2.4) with arithmetic carried out mod M. Let M have the following unique prime power factorization:

$$M = p_1^{r_1} p_2^{r_2} \cdots p_\ell^{r_\ell} \tag{2.11}$$

where the p_i's are distinct primes.

As pointed out previously (section II), when one carries arithmetic mod M, one is in effect doing it modulo each $p_i^{r_i}$ simultaneously. Therefore, the length N number theoretic transform, having the cyclic convolution property in Z_M, must also have that property in

$$Z_{p_i^{r_i}} \qquad \text{for} \quad i = 1, 2, \ldots \ell.$$

This requires that $\alpha (\text{mod } p_i^{r_i})$, an integer of order N must exist in $Z_{p_i^{r_i}}$, i.e., N is the least positive integer such that

$$\alpha^N = 1 \ (\text{mod } p_i^{r_i}), \quad i = 1,2, \ldots \ell \tag{3.9}$$

Example: Let $M = 24 = 2^3 \times 3$

then

$$Z_{2^3} = Z_8 = \{0,1,2,3,4,5,6,7\}$$

and

$$Z_3 = \{0,1,2\}$$

For (3.9) to be satisfied, since

i) for Z_3

N	0	1	2
1^N	1	1	1
2^N	1	2	1

(3.10)

order 2

, i.e., $\alpha = 1$ order $N = 1$

$\alpha = 2$ order $N = 2$

and

ii) for Z_8

N	0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1	1
2	1	2	4	0	0	0	0	0

(3.11)

i.e., $\alpha = 1$ order $N = 1$

Then, $\alpha = 1 \pmod{24}$ is a root of order $N = 1$, thus the length of the number theoretic transform possible in the ring Z_{24}, is $N = 1$ (not a very interesting result!). Furthermore, since the inverse transform requires the existence of the inverse of N, i.e., N^{-1}, this number should exist in $Z_{p_i^{r_i}}$, or N should be relatively prime to M (2.21). In this example,

(i) for Z_3

one wants $N^{-1} = 1^{-1} = ?$

By (2.21a)

$$N^{-1} = 1^{-1} = 1^{3-2} = 1^{1} = 1 \bmod 3$$

(ii) for Z_8

also required is $N^{-1} = 1^{-1} = ?$

By (2.22)

$$N^{-1} = 1^{-1} \pmod 8 = 1^{\phi(M)-1}$$

but

$$\phi(M=8) = 8(1 - \tfrac{1}{2}) = 4$$

then

$$N^{-1} = 1^{-1} \pmod 8 = 1^{4-1} = 1^{3} = 1 \pmod 8$$

Thus we have verified the existence of a number theoretic transform of length N = 1, in the ring of integers Z_{24}. In general, the existence of an of order N, each $Z_{p_i^{r_i}}$ can be investigated recalling Euler's theorem (2.13). That is, by observing (3.4) and Euler's theorem, one has the condition

$$N \mid \phi(p_i^{r_i}) \qquad i = 1, 2, \ldots, \tag{3.12}$$

i.e., N should divide $\phi(p_i^{r_i})$.

Example: Let $M = 11 \times 31 = 341$

Then

$$Z_{11} = \{0,1,2, \ldots, 10\}$$

and

$$Z_{31} = \{0,1,2, \ldots, 30\}$$

i) for Z_{11}

$\phi(11) = 11 - 1 = 10$, implies possible values of order N for the roots in Z_{11} (1,2,5,10)

ii) for Z_{31}

$\phi(31) = 31 - 1 = 30$, implies possible values of order N for the roots in Z_{31} (1,2,3,5,6,10,30)

That is, for (3.12) to be satisfied, in the conditions given by i) and ii), the possible values for the length of the NTT in Z_{341} are (1,2,5,10).

Notice that by (3.9), whatever the root in consideration, it has to be the same order in Z_{11} as that in Z_{31}.

For Z_{11}, one constructs the following table.

N	0	1	2	3	4	5	6	7	8	9	10	
1^N	1	1	1	1	1	1	1	1	1	1	1	root of order 1
2^N	1	2	4	8	5	10	9	7	3	6	1	" " " 10
3^N	1	3	9	5	4	1	3	9	5	4	1	" " " 5 (3.12a)
4^N	1	4	5	9	3	1	4	5	9	3	1	" " " 5
5^N	1	5	3	4	9	1	5	3	4	9	1	" " " 5
6^N	1	6	3	7	9	10	5	8	4	2	1	" " " 10
7^N	1	7	5	2	3	10	4	6	9	8	1	" " " 10
8^N	1	8	9	6	4	10	3	2	5	7	1	" " " 10
9^N	1	9	4	3	5	1	9	4	3	5	1	" " " 5
10^N	1	10	1	10	1	10	1	10	1	10	1	root of order 5

For Z_{31}, a similar table is constructed, from which only a part is shown (notice that only N = 1, 2, 5, 10 are of interest).

N	0	1	2	5	10	
1^N	1	1	1	1	1	root of order 1
2^N	1	2	4	1	1	root of order 5
3^N	1	3	9	26	25	root of order 30
4^N	1	4	16	1	1	root of order 5
5^N	1	5	25			
⋮						
30^N	1	30	1	30		root of order 2

Thus the root $\alpha = 4$ is of order 5 in both Z_{11} and Z_{31}, and so the length of the NTT in the ring Z_{34} can be equal to 5. Furthermore, since the multiplicative inverse of an integer b in Z_M exists if and only if b and M are relatively prime, and for the inverse transform one requires N^{-1}, N should be relatively prime to M (or p_i's).

In the last example:

N = 5 is relative prime to N = 243 so $5^{-1} \bmod 341$ exists and is given by

$$5^{\phi(M)-1} = 5^{\phi(341)-1} \bmod 341$$

but

$$\phi(341) = 341(1 - \frac{1}{10})\ (1 - \frac{1}{31}) = 300$$

so

$$5^{-1} \bmod 341 = 5^{300-1} \bmod 341 = 5^{299} \bmod 341$$

To find this number notice that

$$299 = 256 + 32 + 8 + 2 + 1 = 2^8 + 2^5 + 2^3 + 2^1 + 2^0$$

so

$$5^{299} = 5^{256} \cdot 5^{32} \cdot 5^{8} \cdot 5^{2} \cdot 5$$

But

$$5^2 = 25 \bmod 341$$

$$5^4 = 284 \bmod 341$$

$$5^8 = 180 \bmod 341$$

$$5^{16} = 5 \bmod 341$$

$$5^{32} = 25 \bmod 341$$

$$5^{64} = 284 \bmod 341$$

$$5^{128} = 180 \bmod 341$$

$$5^{256} = 5 \bmod 341$$

Then

$$5^{299} \bmod 341 = (5 \cdot 25 \cdot 180 \cdot 25 \cdot 5) \bmod 341$$

$$= 273 \bmod 341$$

Check:

$$(5 \times 273) \bmod 341 = 1 \bmod 341.$$

So

$$5^{-1} \bmod 341 = 273 \bmod 341.$$

That is, it is verified that a number theoretic transform of length N = 5 exists in Z_{341}. Now, the condition N relative prime to M (or p_i's) means

$$N|(p_i - 1) , \quad i = 1, 2, \ldots, \ell \tag{3.13}$$

i.e.

$$N|\gcd \{p_1-1, p_2-1, \ldots, p_\ell-1\} .$$

O(M) is defined as the greatest common divisor (gcd) of the $(p_i - 1)$

$$O(M) \triangleq \gcd \{p_1-1, p_2-1, \ldots, p_\ell-1\} . \tag{3.14}$$

Therefore

$$N|O(M) \tag{3.15}$$

This last equation gives the necessary condition for the existence of a transform of length N in the ring Z_N with arithmetic carried out an integer M.

Example: Let $M = 11 \times 31 = 341$

$$O(M) = \gcd\{11-1,31-1\} = \gcd\{10,30\} = 10$$

Notive that N = 5 satisfies the condition (3.15) since

$$N = 5 \mid (O(M) = 10.)$$

It remains to be investigated further whether or not a given α of order N = 10 exists in both Z_{11} and Z_{31}.

Now, consider the converse of condition (3.15). If $N \mid O(M)$ or $N \mid \phi(p_i^{r_i})$, then there exists integers $\alpha_i \pmod{p_i^{r_i}}$ or oder N in $Z_{p_i^{r_i}}$ [13].

Using these α_i's one can construct transform $(\text{mod } p_i^{r_i})$ which have the DFT structure (3.7) and are invertible.

Combining these transforms by the Chinese remainder theorem (2.27), one can obtain a transform (mod M) having the cyclic convolution property in Z_M. Alternatively one can combine the α_i's by the Chinese remainder theorem to obtain an α (mod M) of order N in Z_M and construct the final transform using this α. The results will be identical.

Example: Let $M = 5 \times 17 = 85$

$\phi(5) = 5-1 = 4$ possible values of N: 1,2,4

$\phi(17) = 17-1 = 16$ possible values of N: 1,2,4,8,16

If one looks for an α of order 4 in Z_{85}, by constructing tables for Z_5 and Z_{17} as those in (3.12 a,b), one finds

$$\alpha_1 = 2 \bmod 5 \quad \text{(order 4, in } Z_5\text{)}$$

$$\alpha_2 = 4 \bmod 17 \quad \text{(order 4, in } Z_{17}\text{)}$$

Using the Chinese remainder theorem (2.27):

(i) d_k's calculation

$$d_1 = (5 \times 17)/5 = 17 \bmod 5 = 2 \bmod 5$$

$$d_2 = (5 \times 17)/17 = 5 \bmod 17$$

(ii) Calculation of d_i^{-1}'s.

$$d_1^{-1} \triangleq (2 \bmod 5)^{-1} = 2^{5-2} \bmod 5 = 2^3 = 3 \bmod 5 \quad \text{(by 2.21a)}$$

$$d_2^{-1} \triangleq (5 \bmod 17)^{-1} = 5^{M-2} = 5^{17-2} = 5^{15} = 7 \bmod 17 \quad \text{(by 2.21a)}$$

Then (by (2.27)

$$\alpha = ((2 \times 2 \times 3) + (4 \times 5 \times 7)) \bmod 85$$

$$\alpha = (12 + 140) \bmod 85$$

$$\alpha = 152 \bmod 85 = 67 \bmod 85$$

Check:

$$67^4 = \bmod 85$$

Notice that $\alpha = 64 \bmod 5$ is of order 4 in Z_5, and also $\alpha = 64 \bmod 17$ is of order 4 in Z_{17}.

To establish the existence of a NTT in Z_{85}, with length $N = 4$, one has to find the inverse of $N = 4^{-1}$, i.e.,

$$4^{-1} = ?$$

By (2.22)

$$4^{-1} = 4^{\phi(85)-1} \quad \text{and} \quad \phi(85) = 85(1-\tfrac{1}{5})(1-\tfrac{1}{17}) = 64$$

So

$$4^{-1} = 4^{64-1} = 4^{63} \bmod 85.$$

Since

$$63 = 32 + 16 + 8 + 4 + 2 + 1$$

$$4^{63} = 4^{32} \cdot 4^{16} \cdot 4^{8} \cdot 4^{4} \cdot 4^{2} \cdot 4^{1}$$

But

$$4^{1} = 4 \bmod 85$$

$$4^{2} = 16 \bmod 85$$

$$4^{4} = 1 \bmod 85$$

$$4^8 = 1 \bmod 85$$

$$4^{16} = 1 \bmod 85$$

$$4^{32} = 1 \bmod 85$$

so

$$4^{-1} = 4^{63} = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 16 \cdot 4 = 64 \bmod 85$$

Check:

$$4 \times 64 = 1 \bmod 85.$$

The fact that condition (3.15) holds as well as its converse [13] means that (3.15) is the necessary and sufficient condition for the existence of an invertible transform of length N which has the cyclic convolution property, mod M.

This also establishes that the maximum transform length in Z_M is

$$N_{max} = O(M) \tag{3.16}$$

Notice that for a given modulus one knows exactly what are the possible transform lengths in Z_M.

For any NTT to be computationally efficient, there are three main requirements [17]:

(i) N, the transform length, should be highly composite (preferably a power of 2) for an FFT-type algorithm to exist, and should be large enough for practical sequence lengths.

(ii) The multiplication by powers of α (3.7) must be a a simple operation. This is possible if the powers of α have a binary representation with few bits.

(iii) To simplify the arithmetic modulo M, M should have a binary representation with very few bits and be large enough to prevent overflow.

Although the class of all possible number theoretic transforms seems very large at first consideration, closer examination shows that very few seem to satisfy the aforementioned criteria. The parameters that must be chosen are M, N and α.

Briefly, one verifies that if M is even, it has a factor of 2 and, therefore, O(M) and N_{max} are 1, which implies M should be odd. If M is prime then O(M) = M-1 which is as large as one could hope for in a field of M integers.

In section IV, the various types of NTT are discussed, but one can say that conditions ((3.8),(3.16)) do not give a systematic way of determining the "best" choices. As a result one must use intuition, insight and a bit of searching. Usually M is selected and the resulting possible N and α are then examined.

IV. NUMBER THEORETIC TRANSFORMS

One computational need in the digital processing of signals is the evaluation of the circular convolution summation of two sequences of length N. That is, the evaluation of

$$y(n) = \sum_{m=0}^{N-1} x(n-m)\, h(m) \qquad (4.1)$$

$$n = 0, 1, \ldots N-1$$

The so-called fast convolution procedure obtains this sum by taking the inverse transform of the product of the transforms of the two sequences. If the transform used is the discrete Fourier transform, then error-free results are obtained only if infinite precision arithmetic is assumed. This is true, even if both sequences are composed of finite precision numbers because the Fourier transform involves the irrational number $\exp[-j(2\pi/N)]$.

One way to avoid the round-off errors induced by the transform is to make use of the fast transforms over the finite field [14].

By working in a finite field or ring of integers with arithmetic carried out modulo an integer M, a large class of transforms exist that have the cyclic convolution property (3.5). By special choices of the length N, the mod M, and

the value of α, it is possible to have transforms that need only word shifts and additions but no multiplications, that have an FFT type fast algorithm, that do not require storage of complex values of α, and that have no roundoff errors. These transforms are the Number Theoretic Transforms (NTT) and they look very promising in the evaluation of finite convolutions. Their main disadvantage seems to be a relation of the sequence length N to the required word length that can require long word lengths for long sequences.

This section begins by discussing Mersenne and Fermat number transforms, that proceeds historically all subsequent work in this field. In part B, other number theoretic transforms that provide more choices of word lengths and transform lengths than Mersenne and Fermat number transforms are discussed.

A. MERSENNE AND FERMAT NUMBER TRANSFORMS

Rader [3] suggests performing the calculations of a transform with the DFT structure (3.7), in the ring of integers modulo a Mersenne number. Such numbers are defined by [3]

$$p = 2^q - 1 \tag{4.2}$$

where q is prime, but p is not necessarily prime.

In the ring $p = 2^q-1$, $\alpha = 2$ is a root of the q^{th} order since

$$\alpha^q = 2^q = (2^q - 2^q + 1) = 1 \bmod 2^q - 1 \qquad (4.3)$$

Under these conditions, a Mersenne transform of an integer sequence $\{a_n\}$ having q terms is defined by [3]

$$A_k = \left(\sum_{n=0}^{q-1} a_n \, 2^{nK} \right) \bmod p \qquad (4.4)$$

$$K = 0,1 \ldots q \; .$$

Because q has an inverse modulo p [3], the inverse Mersenne transform will be

$$a_m = \left(q^{-1} \sum_{K=0}^{q-1} A_k \, 2^{-mK} \right) \bmod p \qquad (4.5)$$

$$m = 0,1 \ldots q-1$$

where all exponents and indices being taken modulo q and all operations being performed modulo p in both (4.4) and (4.5).

It can be demonstrated [3] that the Mersenne transform satisfies the convolution theorem; that is to say, if $\{X_K\}$ is the Mersenne transform of $\{x_n\}$, then with $Z_k = A_K \cdot X_K \bmod p$, the Inverse Mersenne transform $\{Z_m\}$ of $\{Z_K\}$ is given by

$$Z_m = \left(\sum_{n=0}^{q-1} a_n \, x_{m-n} \right) \bmod p \qquad (4.6)$$

If $\{a_n\}$ and $\{x_n\}$ are properly bounded [3], Z_m is equal to the output of the ordinary cyclic convolution with

$$Z_m = \sum_{n=0}^{q-1} a_n \, x_{m-n} \qquad (4.7)$$

Under these conditions, digital filtering of real integer sequences can be performed by dividing the sequences into blocks, padding the blocks with zeros [34] to prevent folding, and aliasing and computing the cyclic convolutions by means of Mersenne transforms.

The number of transform terms can be extended to 2q, since $\alpha = -2 \bmod p$ is a root of order 2q:

$$\alpha^{2q} = -2^{2q} = (-2^q)^2 = (-2^q + 2^q - 1)^2 = (-1)^2 = 1 \bmod 2q-1 \qquad (4.8)$$

Example: Let q = 7

Then

$$p = 2^q-1 = 2^7-1 = 127$$

and

$$\alpha = -2 \bmod 127 = (-2 + 127) = 125 \bmod 127.$$

Notice that 2q = 14, so

$$\alpha^{2q} = \alpha^{14} = (-2)^{14} = 125^{14} \bmod 127$$

Now,

$$14 = 2^3 + 2^2 + 2^1 = 8 + 4 + 2$$

Thus

$$125^{14} = 125^8 \cdot 125^4 \cdot 125^2$$

but

$$125^2 = 4 \bmod 127$$

$$125^4 = 16 \bmod 127$$

$$125^8 = 2 \bmod 127$$

and

$$125^{14} = 2 \cdot 16 \cdot 4 \bmod 127$$

$$= 128 \bmod 127$$

$$125^{14} = 1 \bmod 127.$$

Notice also that the inverse of $2q = 14$, mod 127 is

$$\alpha^{-1} = 14^{-1} \bmod 127 = 14^{127 \cdot 2} = 14^{125} \bmod 127 \quad \text{(by 2.21a)}.$$

By performing some simple calculations,

$$14^{-1} = 118 \bmod 127. \quad \text{Check: } 14 \times 14^{-1} = 14 \times 118 = 1 \bmod 127.$$

Thus, a Mersenne transform exists which has the cyclic convolution property for sequences of length N = 2q, with $\alpha = -2$ replacing $\exp[-2\pi j/N]$ as the Nth primitive root of unity in (3.3), and with all calculations in (3.7) done in arithmetic modulo p.

Rader advocated such a transform since using $\alpha = 2$ or $\alpha = -2$ as a root of unity would necessitate only shift and add operations in computing the transform (3.7).

Because Mersenne transforms are evaluated without multiplications, computation of a time-invariant circular convolution (4.1) having N = q points reduces to one multiplication per output sample, as opposed to q multiplications with direct calculation. To compute a FFT of a sequence of length N = q requires of the order of $(N/2) \log_2(N/2)$ complex multiplications [17].

The main limitations of Mersenne transform approach are related to the fact that the number of transforms terms q (or 2q) is not highly composite, since q is a prime. This means that calculations of the transforms cannot be simplified by an FFT-type algorithm.

If one considers $p = 2^K + 1$ and K odd, 3 divides $(2^K + 1)$ and the largest possible transform is 2, thus one considers only K even.

Let $K = s\,2^t$, s odd, t an integer. Then since $p = 2^K + 1$, one has

$$\frac{2^{s2^t} + 1}{2^{2^t} + 1} = n\ , \quad n \text{ an integer.} \tag{4.9}$$

And the length of possible transforms will be governed by the length possible for $2^{2^t}+1$ (see, 3.15). Therefore integers of the form

$$p = 2^{2^t} + 1 \tag{4.11}$$

$$t = 0, 1, 2, 3 \ldots$$

are of interest. These numbers are called Fermat numbers and are defined by $F_t = p$ in (4.11). For $t = 0$ to $t = 4$ the Fermat numbers are prime. For $t > 4$ there are no known Fermat number prime.

Number theoretic transforms with a Fermat number as the modulus, are called Fermat number transforms (FNT).

By (3.15), for the FNT of length N to exist N must divide 0 ($F_t = p$).

Notice that (by 3.16), for F_t prime

$$N_{max} = 0(F_t) = 2^b\ , \quad b = 2^t \tag{4.12}$$

and one can have FNT, for any length

$$N = 2^m, \qquad m \leq b \tag{4.13}$$

Example: Let $p = F_2 = 2^{2^2}+1 = 2^4 + 1 = 17$

If one constructs the following table:

	N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
order 1	1^N	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
order 8	2^N	1	2	4	8	16	15	13	9	1	2	4	8	16	15	13	9	1
order 16	3^N	1	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1
order 4	4^N	1	4	16	13	1	4	16	13	1	4	16	13	1	4	16	13	1
order 16	5^N	1	5	8	6	13	14	2	10	16	12	9	11	4	3	15	7	1
order 16	6^N	1	6	2	12	4	7	8	14	16	11	15	5	13	10	9	3	1
order 16	7^N	1	7	15	3	4	11	9	12	16	10	2	14	13	6	8	5	1
order 8	8^N	1	8	12	5	16	9	4	15	1	8	12	5	16	9	4	15	1
order 8	9^N	1	9	3	15	16	8	4	2	1	9	3	15	16	8	4	2	1
order 16	10^N	1	10	15	14	4	6	9	5	16	7	2	3	13	11	8	12	1
order 16	11^N	1	11	2	5	4	10	8	3	16	6	15	12	13	7	9	14	1
order 16	12^N	1	12	8	11	13	3	2	7	16	5	9	6	4	14	15	10	1
order 4	13^N	1	13	16	4	1	13	16	4	1	13	16	4	1	13	16	4	1
order 16	14^N	1	14	9	7	13	12	15	6	16	3	8	10	4	5	2	11	1
order 8	15^N	1	15	4	9	16	2	13	8	1	15	4	9	16	2	13	8	1
order 2	16^N	1	16	1	16	1	16	1	16	1	16	1	16	1	16	1	16	1

(4.13a)

one sees that (3,5,6,7,10,11,12,14) are primitive roots that will generate the entire field Z_{17}. The root $\alpha = 2$ is of order 8 and $\alpha^2 = 2^2 = 4$ is of order $8/2 = 4$ (by 2.20). Also note that $11 = \sqrt{2}$, in the sense that $11^2 = 2 \bmod 17$.

That is, for the ring Z_{17}, one has the possibility of choosing a FNT of lengths (1,2,4,8,16) thus satisfying (4.12) and (4.13).

For digital filtering applications the composites F_5 (b = 32) and F_6 (b = 64) seem to be practical [4]. Lucas [6] has proven that every prime factor of a composite F_t, is of the form

$$K\, 2^{t+2} + 1 \qquad (4.14)$$

Therefore, 2^{t+2} divides 0 (F_t), for $t > 4$.

Example: Consider $F_5 = 2^{2^5} + 1 = 4\ 294\ 967\ 297$

$= 641 \times 6\ 700\ 417$

$$0(F_5) = \text{g.c.d.}\{(641-1),(6700\ 517-1)\}$$

by

$$0(F_5) = \text{g.c.d.}\{640,\ 6\ 700\ 416\}$$

$$640 = 2^7 \cdot 5$$

$$6\ 700\ 416 = 2^7 \cdot 3 \cdot 17449$$

Then

$$0\ (F_5) \;=\; 2^7 \;=\; 128 \;=\; 2^{5+2} \quad \text{where} \quad t = 5.$$

Therefore, for the choice of Fermat numbers with $t > 4$, the maximum possible transform length is given by (see, 3.16)

$$N_{max} \;=\; 0\ (F_t) \;=\; 2^{t+2} \;=\; 2^2 \cdot 2^t \;=\; 4b \qquad (4.15)$$

where

$$t > 4, \quad b \;=\; 2^t .$$

Agarwal [4] proved that

$$\alpha \;=\; 2^{b/4}(2^{b/2} - 1) \qquad (4.16)$$

is a root of order 4b, in Z_{F_t}, $t \geq 2$.

Notice that

$$\alpha^2 \;=\; [2^{b/4}(2^{b/2} - 1)]^2 \;=\; 2^{b/2}(2^{b/2} - 1)^2$$

$$\alpha^2 \;=\; 2^{b/2}(2^b - 2\cdot 2^{b/2} + 1) \;=\; 2^{b/2}(-2 \cdot 2^{b/2})$$

$$\alpha^2 \;=\; (-2)\ 2^b$$

Thus

$$\alpha^2 = 2 \bmod 2^b + 1.$$

Since $\alpha^2 = 2 \bmod F_t$, the notation $\alpha = \sqrt{2}$ for (4.16) will be adopted in this thesis following a general procedure.

Also, note that any odd power of $\sqrt{2}$ will also be of order 2^{t+2} (by 2.20), i.e.

$$\alpha^d = \sqrt{2}^d, \quad d \text{ odd} \tag{4.17}$$

is a root of order 2^{t+2}. And raising $\alpha = \sqrt{2}$ (4.16) to $2^{(t+2-m)^{th}}$ power one obtains an integer α' of order 2^m, $m \leq t+2$, i.e.

$$\alpha^{2^{(t+2-m)}} = \alpha' \tag{4.18}$$

α' being a root of order 2^m, $m \leq t+2$.

Example: Let $p = F_2 = 2^{2^2}+1 = 2^4+1 = 17$

i.e.

$$b = 2^t = 2^2 = 4$$

Then

$$\alpha = 2^{b/4}(2^{b/2}-1) = 2^{4/4}(2^{4/2}-1) = 2(2^2-1)$$

$$\alpha = 2(3) = 6 \bmod 17$$

is a root of order $4b = 4 \cdot 4 = 16$. Observing (4.13a) one sees that this result is correct.

Notice also in (4.13a) that $6 = \sqrt{2}$ in the sense that $6^2 = 2 \bmod 17$. So

$$\alpha = \sqrt{2} = 6 \bmod 17.$$

If one raises this α to $2^{(t+2-m)}$, $m = 3 \leq t+2 = 4$ i.e.

$$2^{(2+2-3)} = 2$$

$$\alpha^2 = 6^2 = 2 \bmod 17$$

and $\alpha = 2$ is a root of order $2^m = 2^3 = 8$ (4.18)

Observing (4.13a) one verifies that this is also a correct result.

If one raises $\alpha = 6$ to an odd power say $d = 5$, $\alpha^5 = 6^5 = 7 \bmod 17$ is a root of the same order as $\alpha = 6$, namely of order $4b = 16$, verifying (4.17). Observing (4.13a) one verifies that this is correct.

Notice that $\alpha = 2 \bmod (2^b+1)$ is of order $N = 2b$, since

$$2^{2b} = 2^{2 \cdot 2^t} = 2^{2^{t+1}}$$

but

$$2^{2b} = (2^b)^2 = (-1)^2 = 1 \bmod (2^b + 1) \qquad (4.19)$$

Thus for FNT's with a prime or composite modulus one sees that $\alpha = 2$ (or a power of 2) is a possible root of order $N = 2b = 2 \cdot 2^t = 2^{t+1}$.

This means that sequences up to a length $N = 2b$, in this case, can make use of FNT.

This is a very desirable situation, since $N = 2^{t+1}$ is highly composite allowing an FFT type algorithm and all multiplication by powers of α are simple word shifts.

If $\alpha = \sqrt{2}$ is used then sequences of length $N = 4b = 2^{t+2}$ are possible (4.16). Recalling that Fermat numbers up to F_4 are prime $O(F(t)) = 2^b$ (by 3.14), and in these cases one can have an FNT for any length $N = 2^m$, $m \leq b$.

Notice that, for these Fermat numbers, $\alpha = 3$, is a root of order $N = 2^b$ (see 4.13a), allowing the largest possible length, in the correspondnet ring Z_{F_t}.

The following table shows some parameters for several possible implementations for FNT's:

t	$b = 2^t$	$F_t = 2^b+1$	$N = 2b$ ($\alpha = 2$)	$N = 4b$ ($\alpha = \sqrt{2}$)	N_{max}	α for N_{max}
2	4	$2^4 + 1$	8	16	16	3,5,6,7, 10,11,12,14
3	8	$2^8 + 1$	16	32	$256=2^b$	3
4	16	$2^{16} + 1$	32	64	$65536=2^b$	3
5	32	$2^{32} + 1$	64	128	128	$\sqrt{2}$
6	64	$2^{64} + 1$	128	256	256	$\sqrt{2}$

That is, $\alpha = \sqrt{2}$ and the resulting $N = 4b$ give the maximum length possible for F_5 and F_6. However, for prime F_t, further increases in N are possible up to $N = 2^b$ if more stages of the FFT algorithm are allowed to have multiplications rather than simple word shifts.

Notice also that besides $\alpha = 3$ being of order $N = 2^b$ there are $2^{b-1} - 1$ other integers of order 2^b, since:

$$\phi(M) = (2^b + 1) - 1 = 2^b,$$

and the number of primitive roots in Z_{F_t} is given by (see section II)

$$\phi(\phi(M)) = 2^b(1 - \tfrac{1}{2}) = 2^{b-1} .$$

For $F_2 = 17$, one sees that the number of primitive roots is

$$2^{4-1} = 8 \quad (3,5,6,7,10,11,12,14) .$$

The cyclic nature of modular arithmetic means that, without a priori knowledge, integers cannot be associated with magnitude. For example, the days of the week represent a modulo 7 system, so that the statement "Friday is after Wednesday" has no meaning unless the week for the Friday and Wednesday in question is specified.

This means that any number theoretic transform unlike the DFT has no physical significance like "frequency." It is merely a transform space, the halfway stage in convolution.

During various stages of the computation of an NTT, each accumulation of signal "overflows" many times.

But still the end result of the convolution will be exact if the input signals are properly bounded [17]. That is, a dynamic range constraint is imposed by the modular arithmetic. One must be able to bound a priori the result of convolution in order to determine the true answer from the answer modulo F_t.

The worst case bound is determined by the following procedure. If

$$c_n = a_n \textcircled{N} b_n$$

where $\textcircled{N}$ means circular convolution of length N, and

$$\begin{aligned} |a_n| &\leq 2^{b_a} \\ |b_n| &\leq 2^{b_b} \end{aligned} \tag{4.21}$$

then

$$|c_n| \leq N\, 2^{b_a+b_b} \tag{4.22}$$

Example:

The circular convolution of two sequences requiring 8 bits plus sign, will require at most, 22 bits plus sign to represent the output sequence. Since

$$N = 64 = 2^6 \qquad |c_n| \leq 2^6 \cdot 2^{8+8}$$

$$b_a = b_b = 8 \qquad |c_n| \leq 2^6 \cdot 2^{16}$$

$$|c_n| \leq 2^{22}$$

Arithmetic modulo F_t can be implemented using $b = 2^t$ bit representation of integers with some provision for representing 2^b.

Section V deals with the implementation of Fermat Number Transforms, where arithmetic is carried modulo $F_t = 2^b + 1$, $b = 2^t$.

Notice that the maximum length of sequences which can be cycled convolved using the FNT with $\alpha = 2$ is $N = 2b$ ($N = 4b$ for $\alpha = \sqrt{2}$), and therefore the length of sequences which can be convoled is proportional to the word length in bits (b).

Thus for long sequences the word length requirement may be excessive.

Rader [3] suggested using a two dimensional convolution scheme to convolve long one dimensional sequences and Agarwal and Burrus [17, [18] presented such a two dimensional

convolution scheme. Using this scheme, cyclic convolution of length N = LP is implemented as a two dimensional cyclic convolution of length 2LXP.

This two dimensional cyclic convolution can be implemented using a two-dimensional FNT. Then the word length required is proportional to the square root of the length of the sequences to be convolved [13], which would give for a maximum sequence length $8b^2$ rather than 4b (for $\alpha = \sqrt{2}$), ie., for a computer's word length b = 64, the maximum length for the transform will be

$$N_{max} = 8b^2 = 32\ 768.$$

Appendix A illustrates with an example such a scheme and also several other points: treatment of negative values in data, the structure of the transform and the inverse matrix, negative powers of α, frequent "overflow" during computations, meaninglessness of the transform values and exactness of the final answer. This example will not demonstrate the efficient implementation of the FNT using the binary arithmetic.

B. OTHER NUMBER THEORETIC TRANSFORMS

Mersenne and Fermat number transforms are very promising for digital filter computation because they can be calculated without multiplications. Their main drawback is a rigid relationship between transform length and wordlength,

caused by the fact that all operations are performed in a finite ring with arithmetic carried out an integer M. Another difficulty arises because it is not possible to achieve simultaneously optimum efficiency in reducing the number of operations and in implementing arithmetic operations. This is so because Fermat number transforms are amenable to a fast transform algorithm, and Mersenne transforms are not, whereas arithmetic operations can be implemented more efficiently modulo a Mersenne number than modulo a Fermat number [3], [13].

In what follows, other number theoretic transforms will be described briefly. Such transforms provide more choices of word length and transform length, thus enlarging the possibility of the use of NTT in digital filtering.

1. Transforms Over the Galois Field $GF(p^2)$

Reed and Truong [20] generalized the ideas of number theoretic transforms to transforms over the Galois Field $GF(p^2)$ where p is a prime Mersenne number, i.e.

$$p = 2^q - 1 , \quad p = 2, 3, 7, 13, 19, 31, 61 \ldots \tag{4.23}$$

Notice that this is a particular case of the definition (4.2), in the sense that here one is interested only in p, a prime. Also, the Galois field $GF(p^2)$ is a particular case of the more general $GF(p^n)$, where one is interested in the case $n = 2$.

Let $GF(p^2)$ denote the Galois Field (finite field) of p^2 elements, where $p = 2^q - 1$, p and q primes. Let d be a divisor of $p^2 - 1$ (possibly $d = p^2 - 1$). Also let the element $r \varepsilon GF(p^2)$, generate the cyclic subgroup of d elements,

$$G_d = (r, r^2 \ldots r^{d-1}, 1) \tag{4.24}$$

Then a transform over this subgroup G_d can be defined by the following pair [20]

$$A_K = \sum_{n=0}^{d-1} a_n r^{Kn}, \quad \text{for} \quad 0 \leq K \leq d-1 \tag{4.25a}$$

and

$$a_m = (d)^{-1} \sum_{K=0}^{d-1} A_k r^{-Km}, \quad \text{for } 0 \leq m \leq d-1 \tag{4.25b}$$

where d divides $p^2 - 1$, a_m and A_k are elements of $GF(p^2)$ and r is a generator of the element subgroup G_d.

It can be shown that the cyclic convolution property holds for this transform [20].

Now, if

$$x^2 = -1 \bmod p \tag{4.26}$$

is not solvable, then the nonsolvability of (4.26) is equivalent to the statement:

(-1) is a quadratic nonresidue mod p

(see Appendix B, eq. B-2).

By Euler's criterion (Appendix B, theor. B.3), this is further equivalent to:

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = (-1)^{[(2^q-1)-1]/2}$$

$$= (-1)^{(2^q-2)/2} \qquad (4.27)$$

$$= (-1)^{(2^q-1)} = -1$$

where

$\left(\frac{a}{p}\right)$ is the Legendre symbol defined by [see Appendix B, eq. (B.8)]

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a \text{ is a quadratic residue mod } p \\ -1 & \text{if } a \text{ is a quadratic nonresidue mod } p \end{cases}$$

Thus (-1) is a quadratic nonresidue mod p and (4.26) is not solvable in GF(p); the polynomial

$$p(x) = x^2 + 1 \qquad (4.28)$$

is then irreducible in GF(p). A root say, $\hat{i}$, of

$$p(x) = x^2 + 1 = 0 \tag{4.29}$$

exists and can be found in the extension field $GF(p^2)$ [20]. The Galois field $GF(p^2)$ is composed of the set

$$GF(p^2) = \{a + \hat{i}b \mid a,b \in GF(p)\} \tag{4.30}$$

where $\hat{i}$ is a root of (4.29), satisfying

$$\hat{i}^2 = -1 \tag{4.31}$$

where $-1 \equiv (p-1) \bmod p$.

If $x^2 + 1 = 0$ is not solvable in GF(p), $\hat{i} \in GF(p^2)$ plays a similar role over the finite field GF(p) that $\sqrt{-1} = i$ plays over the field of rational numbers.

For example, suppose $(a + \hat{i}b)$ and $(c + \hat{i}d)$ are elements of $GF(p^2)$, then by (4.31)

$$\text{(i)} \quad (a + \hat{i}b) \pm (c + \hat{i}d) = (a \pm c) + \hat{i}(b \pm d) \tag{4.32}$$

$$\text{(ii)} \quad (a + \hat{i}b)(c + \hat{i}d) = ac + \hat{i}^2bd + \hat{i}bc + \hat{i}ad \tag{4.33}$$

$$= (ac - bd) + \hat{i}(bc + ad)$$

These are the analogues of what one might expect if $(a + \hat{i}b)$ and $(c + \hat{i}d)$ were complex numbers. In applications to radar and communication systems one generally wants to take convolutions of complex numbers.

If (-1) is a quadratic nonresidue mod p, then convolution (4.20) of the complex integers can be performed with transforms of type (4.25) on a Galois Field $GF(p^2)$ where a_n and b_n are restricted to $GF(p^2)$. In other words, if $a_n, b_n \varepsilon GF(p^2)$, for n = 0,1,2, ... d-1 the transforms are

$$A_K = \sum_{n=0}^{d-1} a_n r^{Kn}$$

$$B_K = \sum_{n=0}^{d-1} b_n r^{Kn} \quad , \quad \text{for} \quad K = 0,1, \ldots d-1.$$

where r is a generator of a d-element subgroup G_d.

Then taking the inverse transform of

$$C_K = A_K \cdot B_K \ , \quad \text{for } K = 0,1, \ldots d-1 \qquad (4.34)$$

one obtains

$$c_n = (d)^{-1} \sum_{K=0}^{d-1} C_K r^{-Kn} \qquad (4.35)$$

where $c_n \varepsilon GF(p^2)$ and $\underline{d}$ divides $p^2 - 1$.

If p is a Mersenne prime, the order t of the subgroup with generator α, factor as follows [20]:

$$\begin{aligned} t = (2^{\hat{q}}-1)^2 - 1 &= 2^{2\hat{q}} - 2\cdot 2^{\hat{q}} + 1 - 1 \\ &= 2^{2q} - 2^{q+1} + 1 - 1 \qquad (4.36) \\ &= 2^{q+1}(2^{2q-q-1} - 1) \\ &= 2^{q+1}(2^{q-1} - 1) \end{aligned}$$

Since t has the factor $d = 2^{\hat{q}+1}$, the usual FFT algorithm can be used to calculate transforms of as many $d = 2^{q+1}$ points. If

$$d = 2^K, \quad 1 \leq K \leq q + 1 \qquad (4.37)$$

and α is a primitive element of $GF(p^2)$, then the generator of G_d is

$$r = \alpha^{2^q} \qquad (4.38)$$

In (4.25) if one wants to take the transform over $GF(p^2)$, of 2^{q+1} point sequences of complex integers, $a_n \in GF(p^2)$ then one needs to find a primitive element in $G_{2^{q+1}}$ of $GF(p^2)$. To achieve this, the following theorems are useful. For proofs, see [20].

Theorem 1:

If p is a Mersenne prime and $d = 2^K$, where $1 \leq K \leq q+1$, then $r = a + \hat{i}b$ is a primitive element in G_d of $GF(p^2)$, if and only if

$$r^{d/2} = -1 \bmod p . \tag{4.39}$$

Theorem 2:

For Mersenne primes $p > 3$, the first quadratic nonresidue modulo p in the sequence 1, 2, 3, ..., p-1, is 3.

To find a primitive element in $G_{2^{q+1}}$ of $GF(p^2)$, one can use the following procedure.

Assume an element $r = a + \hat{i}b$ is of order 2^{q+1} in $GF(p^2)$. Now

$$(a + \hat{i}b)^{2^q} = (a + \hat{i}b)\,(a + \hat{i}b)^{2^q-1} \tag{4.40}$$

and, it can be proved [20] that

$$(a + \hat{i}b)^{2^q-1} \equiv (a + \hat{i}^{2^q-1}b) \bmod p . \tag{4.41}$$

Since

$$\hat{i}^{2^q-1} = \hat{i}\,i^{2^q-2} = \hat{i}\,(\hat{i}^2)^{(2^q-2)/2}$$

$$= \hat{i}\,(-1)^{(2^q-2)/2} = -\hat{i} \tag{4.42}$$

Recall that q = 2, 3, 5, 7, 13, 17, 19, 31, 61, ... i.e., prime. So that (4.40) becomes

$$(a + \hat{i}b)^{2^q} \equiv (a + \hat{i}b)(a - \hat{i}b) \bmod p$$

$$\equiv a^2 + b^2 \bmod p \qquad (4.43)$$

By Theorem 1, it follows that

$$(a + \hat{i}b)^{2^q} \equiv a^2 + b^2 \equiv -1 \bmod p \qquad (4.44)$$

since

$$d = 2^{q+1} \quad \text{and} \quad \frac{2^{q+1}}{2} = 2^q .$$

Let

$$X \equiv a^2 \bmod 2^q - 1$$

$$Y \equiv -b^2 \bmod 2^q - 1 \qquad (4.45)$$

then (4.44) becomes

$$X + 1 \equiv Y \bmod p \qquad (4.46)$$

By definition in (4.46), X is a quadratic residue. For Y, [see Appendix B, eq. B.11, and B.9]

$$\left(\frac{Y}{p}\right) = \left(\frac{-b^2}{p}\right) = \left(\frac{-1}{p}\right)\left(\frac{b^2}{p}\right) = \left(\frac{-1}{p}\right)$$

also (Appendix B, corollary B.5)

$$\left(\frac{-1}{p}\right) = (-1)^{(2^q-1-1)/2} = (-1)^{2^{q-1}-1} = -1$$

thus

$$Y = X + 1 \text{ is a quadratic nonresidue.}$$

Hence, one way to choose the numbers X and Y is to let X and Y be two consecutive numbers from the set of integers $1,2, \ldots 2^q-2$, such that the first number X is a square and the second is a nonsquare. By Theorem 2, for $p > 3$, $Y = 3$ is a nonsquare and the preceding element $X = 2$ is a square. Thus it is sufficient to let

$$a^2 \equiv X \equiv 2 \bmod 2^q - 1 \qquad (4.47a)$$

$$b^2 \equiv -X - 1 \equiv -Y \equiv -3 \bmod 2^q - 1 \qquad (4.47b)$$

To find the solution of congruence (4.47a) one uses the following procedure [20].

First, notice that [20]

$$\left(\frac{2}{2^p - 1}\right) = 1 \qquad (4.48)$$

Then by Euler's criterion [Appendix B]

$$\left(\frac{2}{2^q - 1}\right) \equiv 2^{((2^q-1)-1)/2} \equiv 1 \bmod (2^q-1) \quad (4.49)$$

Multiplying both sides of the congruence by 2, then

$$2^{((2^q-2)/2)+1} = 2^{2^{q-1}} \equiv 2 \bmod (2^q-1)$$

Hence

$$(2^{2^{q-2}})^2 \equiv 2 \bmod (2^q-1) \quad (4.50)$$

Then the solution for (4.47a) is

$$a \equiv \pm 2^{2^{q-2}} \bmod (2^q-1) \quad (4.51)$$

Using the same procedure for (4.47b), b is given by [20]

$$b \equiv \pm (-3)^{2^{q-2}} \bmod 2^{q-1} \quad (4.52)$$

In G_d of $GF(p^2)$ there always exists a primitive element, $r = a + \hat{i}b$ [20]. By Theorem 1 (4.39)

$$(a + \hat{i}b)^{d/2} \equiv -1 \bmod p \quad (4.53)$$

Raising both sides of (4.53) to the jth power, (4.53) becomes

$$((a + \hat{i}b)^{d/2})^j \equiv (-1)^j \bmod p \qquad (4.54)$$

By Theorem 1, $((a + \hat{i}b)^{d/2})^j$ is a primitive element only when j is an odd number. The elements are distinct and include all primitive elements of G_d.

In other words, in the cyclic subgroup of G_d of $GF(p^2)$, if $(a + \hat{i}b)$ is a primitive element then $(a + \hat{i}b)^j$ is also a primitive element for j = 1,3,5, ..., d-1.

Assume $\tilde{\alpha}$ is of order 2^{q+1} in $GF(p^2)$. If d divides 2^{q+1} then $r = \alpha^{2^{2+1}/d}$ is of order d in $GF(p^2)$.

For the transform in (4.25), with d a factor of 2^{9+1}, one can take $r = \alpha^{2^{q+1}/d}$ as the primitive element and d as the transform length.

Example: Find all primitive elements expressed as a sum of powers of two in the subgroup G_d of $GF(p^2)$, where $q = 5$, $p = 2^q - 1 = 31$ and $d = 2^3$.

To do this, first find an element with order $2^{q+1} = 2^{5+1} = 64$ in $GF(31^2)$.

According to Theorem 1, if $r = (a + \hat{i}b)$ is a primitive element in G_{2^6} of $GF(31^2)$, then

$$(a + \hat{i}b)^{2^5} \equiv -1 \bmod 31$$

By (4.44), that becomes

$$a^2 + b^2 \equiv -1 \bmod 31$$

By Theorem 2, 3 is a nonsquare and 2 is a square. By (4.51)

$$a \equiv 2^{2^{5-2}} \equiv 2^{2^3} \equiv 2^8 \equiv 8 \bmod 31$$

By (4.52)

$$b \equiv (-3)^{2^{5-2}} \equiv (-3)^{2^3} \equiv (-3)^8 \equiv (28)^8 \equiv 20 \bmod 31$$

Thus 64 is the smallest positive integer such that

$$(8 + \hat{i}\, 20)^{64} \equiv 1 \bmod 31$$

An element with order $2^{q+1}/d = 64/8 = 8$ is

$$(8 + \hat{i}\, 20)^8 = 8^8 + \binom{8}{1} 8^7 (\hat{i}20) + \binom{8}{2} 8^6 (\hat{i}20)^2$$

$$+ \binom{8}{3} 8^5 (\hat{i}20)^3 + \binom{8}{4} 8^4 (\hat{i}20)^4$$

$$+ \binom{8}{5} 8^3 (\hat{i}20)^5 + \binom{8}{6} 8^2 (\hat{i}20)^6$$

$$+ \binom{8}{7} 8 (\hat{i}20)^7 + (\hat{i}20)^8$$

where $\binom{8}{K}$ are the binomial coefficients. Expanding and solving,

$$(8 + \hat{i}\ 20)^8 \equiv 16 + 10\hat{i} - 10 - 19\hat{i} + 9 + 18\hat{i} - 7 - 5\hat{i} + 19$$

$$\equiv (27 + \hat{i}4) \bmod 31$$

and since, if $(a + \hat{i}b)$ is a primitive element in G_d then $(a + \hat{i}b)^j$ is also a primitive element for j - 1,3,5, ... d-1. One has

$$(27 + \hat{i}4)^1 \bmod 31$$

$$(27 + \hat{i}4)^3 \bmod 31 = (4 + \hat{i}4) \bmod 31$$

$$(27 + \hat{i}4)^5 \bmod 31 = (4 + \hat{i}27) \bmod 31$$

$$(27 + \hat{i}4)^7 \bmod 31 = (27 + \hat{i}27) \bmod 31$$

as the primitive elements in G_8 for $GF(31^2)$.

In order to perform multiplications by powers of r in hardware, it might be desirable to represent the q-bit words a and b, where $r = a + \hat{i}b$, as a minimal sum of powers of two. Then, for example, $r^{n+1} \bmod p$ can be obtained by multiplying r^n by r modulo p recursively using a minimal number of bit rotations, q-bit word- complements, and additions [3].

That is, the primitive elements in G_8 of $GF(31^2)$ can be written:

$$(27 + \hat{i}4) = (2^5 - 2^2 - 2^0) + \hat{i}\, 2^2$$

$$(27 + \hat{i}4)^3 \bmod 31 = 4 + \hat{i}4 = 2^2 + \hat{i}\, 2^2$$

$$(27 + \hat{i}4)^5 \bmod 31 = 4 + \hat{i}\, 27 = 2^2 + \hat{i}(2^5 - 2^2 - 2^0)$$

$$(27 + \hat{i}4)^7 \bmod 31 = 27 + \hat{i}27 = (2^5 - 2^2 - 2^0) + \hat{i}(2^5 - 2^2 - 2^0)$$

Notice that, $r = 4 + \hat{i}4$ has the shortest number of terms of power 2. This primitive element is such that multiplications by powers of r in G_8 of $GF(31^2)$ yield the least hardware. Such an r is called the simplest primitive element.

In order to perform the convolution of two d-point sequences of complex integers a_n and b_n with dynamic range A and B, respectively, the components of the circular convolution $c_p = \gamma_p + \hat{i}\, \delta_p$ are required to remain in the interval [20].

$$-\frac{p-1}{2} \leq \gamma_p, \delta_p \leq \frac{p-1}{2} \tag{4.55}$$

Specifically to satisfy (4.55) for all sequences $a_n = \alpha_n + \hat{i}\, \beta_n$ and $b_n = x_n + \hat{i}\, y_n$, such that $|\alpha_n|$, $|\beta_n| \leq A$, and $|x_n|$, $|y_n| \leq B$, it is necessary [20], that:

$$dAB \leq \frac{p-1}{4} \tag{4.52}$$

If A = B, then by (4.52) the largest value of A is

$$A = \left[\sqrt{\frac{p-1}{4d}}\right] \tag{4.53}$$

where [x] denotes the greatest integer less than x. This scaling constraint sometimes forces one to choose an excessive value p in order to avoid overflow. Such a large p implies a computer word length that is often undesirably long.

Example: Let $p = 2^{31} - 1$ and let $d = 2^8$

By (4.53)

$$A = \left[\sqrt{\frac{2^{31} - 2}{4 \times 2^8}}\right] \cong 2^{10}$$

If a_n and b_n are constrained to the interval

$$-2^{10} \leq \alpha_n, \beta_n, x_n, y_n \leq 2^{10}$$

one is guaranteed to keep the components of c_p in the interval

$$-(2^{31}-1)/2 \leq x \leq (2^{31} - 1)/2 .$$

According to [30] and [29], the Chinese Remainder theorem can be employed to develop a ring which is the direct sum of certain Galois fields $GF(p^2)$. This ring is utilized to extend the dynamic range of complex number convolutions.

A disadvantage of this transform is that multiplication by powers of the primitive element is not as simple as that developed by powers of 2, in Mersenne transforms and Fermat number transforms.

Recently, Reed and Liu [36] developed a high-radix FFT algorithm for computing transforms over $GF(p^2)$, where p is a Mersenne prime. This new algorithm requires substantially fewer multiplications than the conventional FFT.

2. Transforms Over the Finite Field GF(p)

Colomb, Reed and Truong [19] introduces a Fourier-like transform in GF(p), where p is a prime of the form $A_n = 3 \times 2^n + 1$. This transform increases the variety of methods and the digital word lengths that can be used for computing the convolutions of integers beyond the previous Fermat or Mersenne number transforms.

Let GF(p) be the finite field of residue classes modulo p, and let the integer d divide p-1. Also let the element $\gamma \in GF(p)$ generate the cyclic subgroup of d elements, G_d of GF(p).

Then a transform over this subgroup G_d can be defined by the pair [19]:

$$A_K = \sum_{n=0}^{d-1} a_n \gamma^{Kn} \qquad 0 \leq K \leq d-1 \qquad (4.54a)$$

$$a_n = (d)^{-1} \sum_{K=0}^{d-1} A_k \gamma^{-Kn} \qquad (4.54b)$$

where a_n, $A_K \in GF(p)$ for $n = 0,1,2, \ldots d-1$ and $(d)^{-1}$ is the inverse of (d) mod p. Also it can be shown [19] that the circular convolution of two finite sequences of integers can be obtained as the inverse transform of the product of the transforms defined by (4.54).

Notice that if p_n is a prime of the form $3 \times 2^n + 1$ the order t of $GF(p_n)$ with generator α, is given by (2.14)

$$t = p_n - 1 = 3 \times 2^n \tag{4.55}$$

Since t has the factor $d = 2^n$, the usual FFT algorithm can be utilized to calculate the transform of as many as $d = 2^n$ points. If $d = 2^K$, $1 \leq K \leq n$ and α is the generator of $GF(p_n)$, then the generator of G_d is (by 2.20)

$$\gamma = \alpha^{3 \cdot 2^n / 2^K} = \alpha^{3 \cdot 2^{n-K}} \tag{4.56}$$

the basic operations used in the transform defined by (4.54) are addition and multiplication mod $3 \times 2^n + 1$.

Algorithms for searching a prime of the form $3 \times 2^n + 1$ have been developed [19], and will not be discussed in this thesis.

As a final remark about this transform one should say that the same type of dynamic range constraint applies here as in transforms over $GF(p^2)$, although in this case GF(p) refers to integer convolutions. A special number theoretic

transform that can be computed using a high radix FFT is defined [23] over GF(p), a finite field of integers modulo primes p of the form $(2^n-1)2^n + 1$. Such primes are special cases of numbers of the form $2^m - 2^n + 1$ proposed by Pollard [22].

If p is a prime of the form $p = (2^n - 1)2^n + 1$ the order t of a primitive root in the finite field GF(p) is given by

$$t = p - 1 = (2^n - 1)2^n \tag{4.57}$$

Since t has a factor 2^n, one can choose $d = 2^\ell$, where $1 \leq \ell \leq n$ as the transform length. The arithmetic in GF(p) with $p = (2^n - 1)2^n + 1$ is similar to the arithmetic in GF(p) where p is a prime of the form $3 \cdot 2^n + 1$. Addition mod p involves at most a triple binary addition [23].

Multiplication by a power of 2 mod p can be implemented by using the combination of table lookups and mod p addition [23]. However, multiplications mod p still needs a full binary multiplication followed by a division [23]. This seems a disadvantage when compared with Fermat number transforms. The FFT over GF(p) is similar to the FFT over the complex number field, except that a root of unity γ in GF(p) replaces $e^{-j2\pi/d}$ and that integer arithmetic modulo p is used.

It turns out [23], that the number of modulo p multiplications required for the evaluation of an FFT over

GF(p), $p = 2^n(2^n-1) + 1$, is greatly reduced when compared to the number of multiplications required to evaluate the FFT over the complex number field.

It was found [23], that the number $A_n = (2^n-1)2^n+1$ is prime only for the cases m = 1,2,4,32. Hence only transforms over GF(p) when $p = (2^{32}-1)2^{32}+1$ have practical application in digital filtering. The maximum transform length in these conditions is equal to 4 294 967 296!

It can be shown that high radix FFT can also be used to compute transforms over a finite ring modulo a number of the form $(2^m-1)2^n+1$ with m even and m = 1 or m = n, in a similar fashion for the GF(p) case. The condition for such a transform to exist was shown in References [22] and [20].

3. Complex Mersenne Transforms and Complex Pseudo-Mersenne Transforms

The main limitations of the Mersenne transform approach are related to the fact that the number of transform terms q is a prime. This means that calculations of the transforms cannot be simplified by an FFT-type algorithm and that the number of transform terms is equal to the word size. These limitations can be slightly alleviated by using a root (-2) instead of 2 in (4.4) and (4.5), as said previously (4.8). The maximum transform length then becomes 2q. It is also possible to increase the maximum convolution size by resorting to multidimensional convolutions [3], [4], [31]. Unfortunately, this result is achieved at the expense of increased requirements for computation and storage.

By defining Complex Mersenne Transforms, it is possible to achieve higher computation efficiency while increasing both maximum transform length and convolution length. Again, in a Merseene ring, with $p = 2^q - 1$, (2) and (-2) are respectively roots of orders q and 2q, corresponding to transforms of lengths q and 2q, respectively.

Since q is a prime, 2^d and -2^d are also roots of q and 2q, provided d is not a multiple of q (by 2.20). Notice that (2j) is a root of order 4q, since

$$(2j)^{4q} = (2^9)^4 \ (j^4)^q = (1)^4 \ (1)^q = 1 \bmod 2^q - 1 \qquad (4.58)$$

Also (1+j) is a root of order 89, since

$$(i+j)^{8q} = [(1 + j)^8]^q$$

and

$$(1+j)^8 = 1 + \binom{8}{1} 1^7 j + \binom{8}{2} 1^6 j^2 + \binom{8}{3} 1 j^3 + \binom{8}{4} 1^4 j^4$$

$$+ \binom{8}{5} 1^3 j^5 + \binom{8}{6} 1^2 j^6 + \binom{8}{7} 1 j^7 + j^8$$

$$= 1 + 8j - 28 - 56j + 70 + 56j - 28 \cdot 8j + 1$$

$$= 16 + 0j$$

Then

$$(1+j)^{8q} = (2^4)^q = (2^q)^4 = (1)^4 \bmod 2^q-1 = 1 \bmod 2^q-1 \quad (4.59)$$

The same conclusions can be drawn with respect to (1-j), ie., (1-j) is a root of order 8q, in the ring $p = 2^q-1$, for q = 2,3,5,7,13,17,19,31,61,...

Higher order complex roots do not have a simple structure and therefore will generally not be of practical interest.

Under these conditions, a Complex Mersenne Transform having 4q terms can be defined by [24],

$$A_K = \sum_{n=0}^{4q-1} a_n \, j^{nK} \, 2^{nK} \bmod (2^q-1) \quad (4.60)$$

$$j = \sqrt{-1} \ , \quad K = 0,1, \ldots, 4q-1$$

Notice that q has an inverse modulo p, and that the inverse of 4 is 2^{q-2}, since

$$4 = 2^2 \quad (4)^{-1} \bmod 2^q-1 = 2^{q-2} \quad (4.61)$$

for

$$2^2 . 2^{q-2} = 2^2 . 2^q . 2^{-2} = 2^q = 1 \bmod (2^q-1).$$

Then 4q has an inverse R such that

$$(4qR) \bmod 2^q - 1 = 1 \bmod 2^q - 1$$

and the inverse transform of A_k is

$$a_m = R \sum_{K=0}^{4q-1} A_K \, j^{-mK} \, 2^{-mK} \bmod 2^q - 1 \qquad (4.62)$$

$$m = 0, 1, \ldots, 4q-1$$

where all exponents and indices are taken, modulo 4q, in both (4.60) and (4.62).

Using a root (j+1) leads to a definition of a Complex Mersenne Transform having 8q terms with

$$A_K = \sum_{n=0}^{8q-1} a_n \, (1+j)^{nK} \bmod 2^q-1 \qquad (4.63)$$

$$K = 0,1, \ldots, 8q-1$$

and, with R such that $(8qR) \bmod 2^q-1 = 1$, an inverse transform

$$a_m = R \sum_{K=0}^{8q-1} A_K \, (1+j)^{-mK} \bmod 2^q-1 \qquad (4.64)$$

$$m = 0, 1, \ldots, 8q-1.$$

with all exponents and indices taken modulo 8q.

Computation of a complex convolution by means of Complex Mersenne Transforms is carried out as with the DFT, with real and complex parts being evaluated modulo p separately. In order to avoid errors due to overflow, the amplitudes of real and imaginary outputs must be bounded to (p-1)/2. This means that usually the word length of real and imaginary parts of input sequencds $\{a_n\}$ and $\{x_n\}$ is less than half that of output sequences. In other words, all computations are carried out modulo p on q-bit words, yielding q-bit word outputs, and the input sequences are represented by words of length less that (q-1)/2 bits. Notice that the calculation of these transforms can be partly simplified by an FFT-type algorithm because the number of terms is no longer a prime.

It has been shown [24] that, in the practical range of interest for q (where q = 31), substituting Complex Mersenne Transforms for conventional Mersenne Transforms results approximately in an eightfold reduction in the number of operations.

If the two sequences $\{y_n\}$ and $\{a_n\}$ to be convolved are real, the full benefit of using Complex Mersenne Transforms can be retained by processing simultaneously two successive blocks of sequence $\{y_n\}$ by means of the same Complex Mersenne Transforms. This is done by computing the complex convolution $\{Z_m\}$, of the sequence $\{a_n\}$ with the auxiliary complex sequence $\{x_n = y_n + j\ y_{n+8q}\}$. The real

part $\{u_m\}$ and the imaginary part $\{\hat{u}_m\}$ of $\{Z_m\}$ yield respectively the convolution of $\{y_n\}$ and the next block $\{y_{n+8q}\}$ with $\{a_n\}$.

Up to now, the discussion of this section has been restricted to Mersenne numbers, that is, to numbers $p = 2^q - 1$ such that p is a prime. If p is not a prime, its prime factorization is given by

$$p = p_1^{d_1} \cdot p_2^{d_2} \cdot \ldots p_i^{d_i} \tag{4.65}$$

Recall that an m-point real transform having the circular convolution property can be defined in the ring of integers modulo p, provided m-point transforms can be defined separately in the fields $p_1, p_2, \ldots, p_i$.

This follows directly from the Chinese remainder theorem (2.27), and leads to the conditions for the existence of an m-point transform in the ring p that m must simultaneously divide $p_1-1, p_2-1, \ldots, p_i-1$. When p is a prime, the maximum length of the transform is $M = p-1$.

Transforms in a ring p, with p nonprime, are therefore proportionally shorter than transforms defined modulo a prime number. If p and q are composites with $q = q_1 \cdot q_2$ and q_1 prime, $2^{q_1}-1$ divides p and the maximum transform length is governed by that possible for $2^{q_1}-1$.

This led Rader [3] to consider that the only transforms of interest in a ring modulo 2^q-1 were Mersenne Transforms.

The situation changes noticeably if one considers Comples Mersenne Transforms. Nussbaumer [24] shows that given an m-point real transform of root 2 with p composite and q, ,m odd integers, one can define 8m-point complex transforms in the ring modulo p with

$$A_K = \sum_{n=0}^{8m-1} a_n (1+j)^{\omega nK}, \qquad \begin{matrix} j = \sqrt{-1} \\ K = 0,1,\ldots,8m-1 \end{matrix} \tag{4.66a}$$

$$a_m = (8m)^{-1} \sum_{K=0}^{8m-1} A_K (1+j)^{-\omega nK}, \quad m = 0,1, \ldots, 8m-1$$

Notice that, the existence of an m-point real transform in the ring modulo p implies that m has an inverse, R modulo p. Also, the inverse of 8 modulo p, is

$$8^{-1} = 2^{q-3} \quad \text{since} \quad 8 \times 8^{-1} = 2^3 \cdot 2^{q-3} = 2^3 \cdot 2^q \cdot 2^{-3} = 2^q = 1 \bmod p$$

thus

$$(8m)^{-1} = 2^{q-3} \cdot R \bmod p \quad \text{exists.}$$

Table I shows the various possibilities for p nonprime and odd q.

The case of q prime (q = 23,29,37,41,43,47) corresponds to conventional Mersenne Transforms. When q is not a

TABLE I

MAXIMUM ODD LENGTH AND CORRESPONDING POWER-OF-2 ROOTS FOR REAL TRANSFORMS MODULO $P = 2^q-1$ WITH q ODD AND p COMPOSITE

q	Prime Factorization of $p = 2^q-1$ $p_1, p_2, \cdots p_i$	Prime factorization of p_i-1	Max. odd length	Power of 2 roots
15	$7 \cdot 31 \cdot 151$	$(2 \cdot 3)\ (2 \cdot 3 \cdot 5)\ (2 \cdot 3 \cdot 5^2)$	3	/
21	$7^2 \cdot 127 \cdot 337$	$(2 \cdot 3)\ (2 \cdot 3^2 \cdot 7)\ (2^4 \cdot 3 \cdot 7)$	3	/
23	$47 \cdot 178481$	$(2 \cdot 23)\ (2^4 \cdot 5 \cdot 23 \cdot 97)$	23	2
25	$31 \cdot 601 \cdot 1801$	$(2 \cdot 3 \cdot 5)\ (2^3 \cdot 3 \cdot 5^2)\ (2^3 \cdot 3^2 \cdot 5^2)$	15	/
27	$7 \cdot 73 \cdot 262657$	$(2 \cdot 3)\ (2^3 \cdot 3^2)\ (2^9 \cdot 3^3 \cdot 19)$	3	/
29	$223 \cdot 1103 \cdot 2089$	$(2^3 \cdot 29)\ (2 \cdot 19 \cdot 29)\ (2^3 \cdot 3^2 \cdot 29)$	29	2
33	$7 \cdot 23 \cdot 89 \cdot 599479$	$(2 \cdot 3)\ (2 \cdot 11)\ (2^3 \cdot 11)\ (2 \cdot 3 \cdot 11 \cdot 31 \cdot 293)$	/	/
35	$31 \cdot 71 \cdot 127 \cdot 122921$	$(2 \cdot 3 \cdot 5)\ (2 \cdot 5 \cdot 7)\ (2 \cdot 3^2 \cdot 7)\ (2^3 \cdot 5 \cdot 7 \cdot 439)$	/	/
37	$223 \cdot 6\ 163\ 18\ 177$	$(2 \cdot 3 \cdot 37)\ (2^5 \cdot 3 \cdot 37, k67, 1039)$	37	2
39	$7 \cdot 79 \cdot 8191 \cdot 121369$	$(2 \cdot 3)\ (2 \cdot 3 \cdot 13)\ (2 \cdot 3^2 \cdot 5 \cdot 7 \cdot 13)$ $\cdot (2^3 \cdot 3 \cdot 13 \cdot 389)$	3	/
41	$13367 \cdot 164511353$	$(2 \cdot 41 \cdot 163)\ (2^3 \cdot 41 \cdot 59 \cdot 8501)$	41	2
43	$431 \cdot 9719 \cdot 2099863$	$(2 \cdot 5 \cdot 43)\ (2 \cdot 43 \cdot 113)\ (2 \cdot 43 \cdot 3^2 \cdot 2713)$	43	2
45	$7 \cdot 31 \cdot 73 \cdot 151 \cdot 631 \cdot 23311$	$(2 \cdot 3)\ (2 \cdot 3 \cdot 5)\ (2^3 \cdot 3^2)\ (2 \cdot 3 \cdot 5^2)$ $\cdot (2 \cdot 3^2 \cdot 5 \cdot 7)\ (2 \cdot 3^2 \cdot 5 \cdot 7 \cdot 37)$	3	/
47	$2351 \cdot 4513 \cdot 13264529$	$(2 \cdot 5^2 \cdot 47)\ (2^5 \cdot 3 \cdot 47)\ (2^4 \cdot 31 \cdot 47 \cdot 569)$	47	2
49	$127 \cdot 4432676798593$	$(2 \cdot 3^2 \cdot 7 (2^7 \cdot 3^2 \cdot 7^2 \cdot 43 \cdot 337 \cdot 5419)$	63	/

prime, the corresponding transforms are called pseudo-Mersenne transforms. They have a very short length and their roots are not powers of 2. Notice that, in order to achieve maximum effectiveness in computing convolutions by means of pseudo-Mersenne Transforms, it would be desirable to have relatively long transforms with a number of terms highly factorizable. This does not seem possible with transforms modulo $p = 2^q-1$. One notes, however that when p is not a prime, with the prime factorization of p defined by (4.65), one can define transforms modulo $p/p_i^{d_i}$ having power of 2 roots and such that the number of terms is large and highly factorizable.

These transforms can be defined by

$$A_K = \left(\sum_{n=0}^{8m-1} a_n (1+j)^{nK} \right) \bmod p/p_i^{d_i} \qquad (4.67)$$

$$K = 0,1 \ldots 8m-1$$

$$j = \sqrt{-1}$$

Various possibilities of such transforms are listed in Table II.

It can be seen that the maximum number of terms is both large (40 to 392 terms) and highly factorizable, thereby leading to efficient FFT type computation with a minimum number of operations. It would seem, however, that these advantages are offset by the fact that the various operations

TABLE II

LENGTH AND ROOTS FOR REAL AND COMPLEX TRANSFORMS IN THE RING $(2^q-1)/p_i^{d_i}$

q	Transform ring	Real transform Length	Real transform Root	Complex transform Length	Complex transform Root	Approximate word length Nb of bits
15	$(2^{15}-1)7$	5	2^3	40 $(2^3 \cdot 5)$	$2(j-1)$	12
21	$(2^{21}-1)/7^2$	7	2^3	56 $(2^3 \cdot 7)$	$2(j-1)$	15
25	$(2^{25}-1)/31$	25	2	200 $(2^3 \cdot 5^2)$	$(j+1)$	20
27	$(2^{27}-1)/7.73$	27	2	216 $(2^3 \cdot 3^3)$	$(j+1)$	18
35	$(2^{35}-1)/31.127$	35	2	280 $(2^3 \cdot 5 \cdot 7)$	$(j+1)$	23
35	$(2^{35}-1)/127$	5	2^7	40 $(2^3 \cdot 5)$	$2^3(1-j)$	28
35	$(2^{35}-1)/31$	7	2^5	56 $(2^3 \cdot 7)$	$-2^2(1+j)$	30
45	$(2^{45}-1)/7.73$	5	2^9	40 $(2^3 \cdot 5)$	$2^4(1+j)$	36
49	$(2^{49}-1)/127$	7	2^7	56 $(2^3 \cdot 7)$	$2^3(1-j)$	42
49	$(2^{49}-1)/127$	49	2	392 $(2^3 \cdot 7^2)$	$(j+1)$	42

are performed modulo $(2^q-1)/p_i^{d_i}$. The corresponding arithmetic circuits are much more complex than arithmetic circuits modulo 2^q-1 [24].

This difficulty can be circumvented by noticing that as

$$p = p_1^{d_1} \cdot p_2^{d_2} \cdots p_i^{d_i} ,$$

one can compute the convolution modulo $p = 2^q-1$ as with conventional Mersenne Transforms and obtain the final result by performing a last operation modulo $p/p_i^{d_i}$ on the convolutions computed modulo p, i.e.,

$$Z_m \bmod p/p_i^{d_i} = (Z_m \bmod p) \bmod p/p_i^{d_i} \qquad (4.68)$$

By proceeding in this fashion relatively long convolutions can be computed efficiently by means of FFT-type algorithms with all but the last operation performed with implemented arithmetic circuits operating modulo (2^q-1) [24].

Taking as an example the case of transforms defined by q = 25, one can see from Table I that the maximum odd length for real transforms computed modulo $p = 2^{25}-1$ is 15 terms and that the corresponding roots are not powers of two. By operating modulo $(2^{25}-1)/31$, it is possible to define real transforms having power-of-two roots with a maximum odd length increased to 25 terms. The maximum

length is then expanded to 200 terms by using complex roots. Such a transform can be computed very efficiently by means of an FFT type algorithm with a three-stage radix 2 decomposition, followed by a two-stage radix 5 decomposition.

One limitation of conventional Mersenne Transforms is the rigid relationship between word length and transform length. In this respect, pseudo-Mersenne Transforms provide a significant improvement because their maximum number of terms M_{max} is highly composite and any transform length submultiple of M_{max} can be selected.

It is even possible to have several transforms of identical length and defined modulo integers p_1, ... p_i that are relatively prime. The convolution can then be computed separately modulo p_1, ... p_i and then the final result obtained modulo $(p_1 \cdot p_2 \cdots p_i)$ by the Chinese remainder theorem. This approach could, for instance, be used to compute a 40-term convolution with an approximate word length of 32 bits by means of transforms defined by modulo $(2^{15}-1)/7$ and $(2^{25}-1)/31$.

4. Pseudo Fermat Number Transforms and Complex Pseudo Fermat Number Transforms

This section considers a generalization of Fermat Number Transforms, such that transforms having roots which are powers of 2 are defined in field or ring which is a factor of an integer, $p = 2^q+1$. With such pseudo Fermat Number transforms it is possible to have much more flexibility

in selecting desired word lengths than with conventional Fermat Number Transforms. In some cases it is possible to define complex pseudo FNT's which are well adapted for filtering complex signals and allow increased transform and convolution lengths when compared to conventional FNT's.

Number theoretic transforms in a ring submultiple of $p = 2^q+1$ are called pseudo Fermat Number Transforms [37].

If one restricts to roots 2 , one can define an M-term pseudo FNT and its inverse by

$$A_K = \left(\sum_{n=0}^{M-1} a_n \, 2^{\omega nK} \right) \bmod p/p_i^{d_i} \qquad (4.69a)$$

$$a_m = \left(R \sum_{K=0}^{M-1} A_K \, 2^{-\omega mK} \right) \bmod p/p_i^{d_i} \qquad (4.69b)$$

$$m = 0,1 \ldots M-1$$

with $M \cdot R = 1 \bmod p.p_i^{d_i}$.

It can be seen that these transforms have the same structure as pseudo Mersenne transforms but are defined in a ring submultiple of 2^q+1 instead of a ring submultiple of 2^q-1 for pseudo Mersenne transforms.

The choice of the particular ring on which pseudo FNT's are defined is very important. Usually, p will be

divided by its smallest factors, with the remaining factors large enough to allow defining long transforms with powers of two roots. Pseudo FNT's can be defined to q even and q odd [37]. Various possibilities for such transforms with q even are listed in Table III.

It can be seen that there is much more flexibility in word length and transform length selection than with transforms defined modulo 2^q+1.

Here also as in the case of pseudo Mersenne transforms, performing the various operations modulo $2^q+1/p_i^{d_i}$ would seem rather awkward, as the corresponding arithmetic circuits are much more complex than those operating mod (2^q+1). Again the solution is to compute the convolution modulo $p = 2^q+1$ as with conventional FNT's and obtain the final result by performing a last operation modulo $p/p_i^{d_i}$ on the convolution evaluated modulo p:

$$Z_m \text{ modulo } p/p_i^{d_i} = (Z_m \text{ modulo } p) \text{ modulo } p/p_i^{d_i} \tag{4.70}$$

Assuming the factorization of the number of transform terms is given by

$$M = M_1 \cdot M_2 \cdots M_j \tag{4.71}$$

the pseudo FNT can be computed by a mixed-radix FFT-type algorithm.

TABLE III

LENGTHS AND ROOTS FOR PSEUDO FNTS IN THE RING $(2^q+1)/p_i^{d_i}$ WITH q EVEN

q	prime factorization of $p = 2^q+1$ $p_1^{d_1}.p_2^{d_2}\cdots p_i^{d_i}$	transform ring	length M	inverse of number of terms R	roots 2^W	approximate output word length Nb of bits
20	17·61681	$(2^{20}+1)/17$	40 $(2^3.5)$	60 139	2	16
22	5·397·2113	$(2^{22}+1)/5$	44 $(2^2.11)$	819 796	2	19
24	97·257·673	$(2^{24}+1)/257$	48 $(2^4.3)$	63921	2	16
26	5·53·157·1613	$(2^{26}+1)/5$	52 $(2^2.13)$	13 163 662	2	24
28	17·15·790·321	$(2^{28}+1)/17$	56 $(2^3.7)$	15 508 351	2	24
34	5·137·953·26 317	$(2^{34}+1)/5$	68 $(2^2.17)$	3 385 444 810	2	32
38	5·229·457·525 313	$(2^{38}+1)/5$	76 $(2^2.19)$	54 252 218 476	2	36
40	257·4 278 255 361	$(2^{40}+1)/257$	80 $(2^4.5)$	4 224 777 169	2	32
44	17·353·2931 542 417	$(2^{44}+1)/17$	88 $(2^3.11)$	1 023 074 990 551	2	40
46	5·277·1013·1657·30269	$(2^{46}+1)/5$	92 $(2^2.23)$	13 920 773 304 712	2	44

If q is odd, it is possible to increase the maximum transform length [37] by using complex pseudo FNT's. The existence of complex pseudo FNT's can be demonstrated by considering an M term pseudo FNT defined in the ring $p/p_i^{d_i} = (2^q+1)/p_i^{d_i}$ with a root 2^W of order M.

Notice that if W and q are odd, the condition

$$M \cdot W = 2q \tag{4.72}$$

implies that M is even, and M/2 is odd. Under these conditions, $(-2)^W$ is a root of order M/2 since

$$[(-2)^W]^{M/2} = (-2)^{WM/2} = (-2)^{2q/2} = (-2)^q$$

$$= (-2^q+2^q+1) = 1 \bmod 2^q+1 \tag{4.73}$$

and $(-2)^{Wd}$ is also a root of order M/2, provided d and q have no common factors [37]. This suggests that $(2j)^W$ is a root of order 2M, since

$$(2j)^{W2M} = (2j)^{4q} = (2^q)^4(j^4)^q = (-1)^4(1)^q = 1 \bmod 2^q+1 \tag{4.74}$$

and $(1+j)^W$ is a root of order 4M since

$$(1+j)^{W4M} = (1+j)^{8q} = (2^4)^q = (2^q)^4 = (-1)^q = 1 \bmod 2^q+1 \tag{4.75}$$

thus a 2M term pseudo FNT can be defined as [37]

$$A_K = \left(\sum_{n=0}^{2M-1} a_n (2j)^{WnK} \right) \bmod p/p_i^{d_i} \qquad \begin{matrix} j = \sqrt{-1} \\ K = 0,1 \ldots 2M-1 \end{matrix} \tag{4.76a}$$

As M has an inverse in the ring $p/p_i^{d_i}$ and 2 is relatively prime with p, 2M has an inverse R in the ring $p/p_i^{d_i}$ and an inverse transform can be defined by [37]

$$a_m = \left(R \sum_{K=0}^{2M-1} A_K (2j)^{-WmK} \right) \bmod p.p_i^{d_i} \qquad (6.76b)$$

$$m = 0,1, \ldots 2M-1$$

with all exponents and indices taken modulo 2M.

It can be demonstrated that the transform satisfies the convolution theorem [37], and that two complex sequences of length 2M can be cyclically convolved via complex pseudo FNT's modulo $p/p_i^{d_i}$.

In such an approach, all arithmetic operations are performed as in normal complex arithmetic with $j^2 = -1$, and real and imaginary parts treated separately modulo p. The final convolution product is obtained by performing a last operation modulo $p/p_i^{d_i}$.

A 4M-points complex transform can be defined by using a root (1+j) instead of (2j) [37]

$$A_K = \left(\sum_{n=0}^{4M-1} a_n (1+j)^{WnK} \right) \bmod p/p_i^{d_i} \qquad (4.77)$$

$$K = 0,1 \ldots 4M-1$$

Higher order complex roots have a complex structure and therefore the maximum length of complex pseudo FNT's which can be computed without multiplications is 4M in the general case, and 8q when W = 1 ($M = \frac{2q}{W}$).

It can be seen that using complex pseudo FNT's allows for a given computation complexity, operation over transforms and convolution lengths twice as long as with real Fermat and pseudo FNT's.

In particular, when W = 1, the maximum length of an FNT is 4q for a root $\sqrt{2}$, while the maximum length of a complex pseudo FNT is 8q for a (1+j) root.

Various possibilities, q odd, for complex pseudo FNT's are listed in Table IV. It can be seen that for q prime, we have complex transforms of length 8q with root (1+j). These transforms have a number of terms which are not highly factorizable.

A more interesting case seems to correspond to those nonprime values of q, for which it is possible to define transforms with a number of terms which are both large and highly factorizable and therefore lead to efficient computation via an FFT type algorithm.

COMPLEX ROOTS AND LENGTHS FOR COMPLEX PSEUDO FNTS IN THE RING $(2^q+1)/p_i^{d_i}$ WITH q ODD

q	prime factorization $p = 2^q+1$ $p_1^{d_1}\cdot p_2^{d_2}\cdots p_i^{d_i}$	transform ring	complex length	inverse of number of terms R	complex roots	approximate output word length Nb of bits
15	$3^2\cdot 11\cdot 331$	$(2^{15}+1)/3^2$	40 $(2^3\cdot 5)$	3 550	2(j-1)	12
21	$3^2\cdot 43\cdot 5419$	$(2^{21}+1)/3^2$	56 $(2^3\cdot 7)$	228 856	2(j-1)	18
25	$3\cdot 11\cdot 251\cdot 4051$	$(2^{25}+1)/3.11$	200 $(2^3.5^2)$	1011 717	j+1	20
27	$3^4\cdot 19\cdot 87\ 211$	$(2^{27}+1)/3^4\cdot 19$	216 $(2^3.3^2)$	21 399	j+1	16
29	$3\cdot 59\cdot 3033\ 169$	$(2^{29}+1)/3$	232 $(2^3.29)$	133 446 362	j+1	27
33	$3^2\cdot 67\cdot 683\cdot 20\ 857$	$(2^{33}+1)3^2$	88 $(2^3.11)$	943 591 300	2(j-1)	30
35	$3\cdot 11\cdot 43\cdot 281\cdot 86\ 171$	$(2^{35}+1)/3\cdot 11$	56 $(2^3\cdot 7)$	1 022 611 261	$-2^2(1+j)$	30
41	$3\cdot 83\cdot 8\ 831\ 418\ 697$	$(2^{41}+1)/3$	328 $(2^3\cdot 41)$	547 521 034 157	(j+1)	39
45	$3^3\cdot 11\cdot 19\cdot 331\cdot 18\ 837001$	$(2^{45}+1)/3^3\cdot 19$	40 $(2^3\cdot 5)$	66 870 882 625	$2^4(1+j)$	36
49	$3\cdot 43\cdot 4\ 363\ 953\ 127\ 297$	$(2^{49}+1)/3\cdot 43$	392 $(2^3.7^2)$	4 352 820 593 809	(j+1)	41

In these respect, the 200-points and 392-points transforms defined respectively modulo $(2^{25}+1)/3.11$ and $(2^{49}+1)/3.43$ are particularly attractive.

It is sometimes desirable to compute convolutions with improved dynamic range. In this case the same convolution can be computed modulo several relatively prime integers p_1, p_2, ... p_i and the final result obtained modulo $(p_1 \cdot p_2 \cdots p_i)$ via the Chinese remainder theorem. For this application, the availability of complex pseudo Mersenne transforms and pseudo FNT's having the same length and defined modulo relatively prime integers is particularly interesting. A 200-point convolution could for instance be computed with a dynamic range of about 40 bits via complex pseudo Mersenne transforms defined modulo $(2^{25}-1)/31$ and via complex pseudo Fermat transforms modulo $(2^{25}+1)/3.11$.

V. IMPLEMENTATION OF FERMAT NUMBER TRANSFORMS

The best known number theoretic transform is the Fermat Number Transform (FNT). FNT are potentially attractive for digital filtering applications because they have the convolution property (3.5) and can be computed without multiplications.

In principle, such Number Theoretic Transforms (NTT's) could be implemented in the same way as Discrete Fourier Transforms but with multiplications by trigonometric functions replaced by multiplications by powers of two, all operations being performed modulo a Fermat number.

In practice, however, direct transposition of Fast Fourier Transform (FFT) architectures does not necessarily lead to optimum implementations and the development of special configurations to computing NTT's seems worth exploring. Along these lines, the special attributes of the FNT, led several researchers to consider various coding techniques for simplifying the implementation of the transform and special purpose implementations of the FNT.

This section will discuss these concepts. In part A various coding techniques are presented and in part B the realization of the FNT is considered.

A. BINARY ARITHMETIC FOR THE FERMAT NUMBER TRANSFORMS

In computing the FNT, arithmetic is done modulo $F_t = 2^b$, $b = 2^t$ (4.11). In this arithmetic the only allowed integers

are $\{0,1 \ldots 2^b\}$ and all integers whose absolute value do not exceed

$$\frac{F_t-1}{2} = \frac{(2^b+1)-1}{2} = \frac{2^b}{2} = 2^{b-1} \qquad (5.1)$$

can be represented unambiguously.

Negative numbers are represented by adding 2^b+1 to them; this is similar to twos complement and ones complement representation of negative integers.

Notice that in a b-bit register, all integers from 0 to 2^b-1 can be represented.

Example: Let $F_t = 2^b+1 = 2^4+1 = 17$ $b = 2^t = 2^2 = 4$

(i) allowed integers $\{0,1,2, \ldots, 16 = 2^4\}$

(ii) absolute values of number that can be represented unambiguously do not exceed

$$\frac{F_t-1}{2} = \frac{17-1}{2} = 8 = 2^{b-1} = 2^{4-1} = 8$$

(iii) negative numbers:

$$-5 = (-5 + 17) = 12 \bmod 17$$

(iv) a b = 4 bit register allows as maximum

$$1111_2 = 8 + 4 + 2 + 1 = 15_{10} = 2^b-1 = 2^4-1 = 15_{10}$$

Thus, using a b-bit register, the problem remains to represent the quantity 2^b (in the example above, $2^b = 2^4 = 16$).

If data is uncorrelated, the probability that this number will appear after an arithmetic operation is approximately 2^{-b} [17].

For digital filter applications, b would typically be 32 or 64; in these cases the occurrence of 2^b is extremely small.

In their software realization of the FNT, Agarwal and Burrus [4] define a binary arithmetic modulo $F_t = 2^b+1$, $b = 2^t$. The representation of such a modulus requires (b+1) bits, for the representation of the quantity $2^b = 1$ mod F_t. In order to simplify modular arithmetic, Agarwal limits his realization to a b-bit arithmetic.

This involves some input quantization error where (-1) occurs as an input sample, as well as the extremely small, but realistic, probability of a complete data block in error when a (-1) occurs as the result of an FNT computation.

The following discussion is based on the b-bit representation of integers. The various basic arithmetical operations can be implemented modulo F_t [4].

(i) Negation

$$\text{Let } A = \sum_{i=0}^{b-1} a_i 2^i , \quad a_i = 0 \text{ or } 1 \qquad (5.2)$$

Then by [4]

$$-A = \sum_{i=0}^{b-1} a_i 2^i = \sum_{i=0}^{b-1} \overline{a_i} 2^i - (2^b-1) \qquad (5.3)$$

Example: Let $A = 8 = (1000)_2 = 8 \bmod 17 \ (= 2^4+1)$.

Then

$$-A = (0111) - (2^4-1) = 7 - (15) = -8$$

Notice that (5.3) can be arranged in the following form:

$$-A = \sum_{i=0}^{b-1} \overline{a_i} 2^i - (2^b-1)$$

$$= \sum_{i=0}^{b-1} \overline{a_i} 2^i - (2^b-1) + (2^b+1) \bmod F_t$$

$$= \sum_{i=0}^{b-1} \overline{a_i} 2^i + 2 \qquad (5.4)$$

Thus to negate a number, one has to complement each bit and add 2 to it.

Example: Let $F_t = 17$

$$A = 8 = 1000 \quad -A = -8 = \begin{array}{r} 0111 \\ \{\underline{+10}\} \\ 1001 \end{array} = 9 \bmod 17$$

Example: Let $F_t = 2^4 + 1 = 17$

and $A = 8 = 1000$; $-8 = \left\{ \begin{array}{r} 0111 \\ +10 \\ \hline 1001 \end{array} \right\} = 9 \bmod 17$

(ii) Addition

When one adds two b-bit integers, one obtains a b-bit integer and possibly a carry bit. The carry bit represents $2^b = -1 \bmod F_t$.

To implement addition in arithmetic modulo 2^b+1, one simply subtracts the carry bit. Thus the hardware should be of the carry subtract type.

Example: Let $F_t = 17$

$$15 + 10 = 25 = 8 \bmod 17 = \left\{ \begin{array}{r} 1111 \\ + \ 1010 \\ \hline 11001 \\ \quad -1 \\ \hline 1000 \end{array} \right\} = 8 \bmod 17$$

(iii) Subtraction

Subtraction is implemented as an addition by first negating the subtrahend and then adding terms. Addition must be carried out according to (ii).

Example: Let $F_t = 17$

$$13 - 4 = 13 + (-4) = \left\{ \begin{array}{l} 13 = 1101 \\ -4 = (\rightarrow 0100 \rightarrow 1011 \\ \qquad\qquad\quad +10 \\ \qquad\qquad\quad \overline{1101} \end{array} \right.$$

$$13 + (-4) = 9 = \begin{cases} 1101 \\ \underline{1101} \\ 11010 \\ \quad\ \ -1 \\ \hline 1001 \end{cases} = 9 \bmod 17$$

(iv) General multiplication

When one multiplies two b-bit integers, one gets a 2b-bit product. Let C_L be the b-bit low order of it and C_H be the b-bit high order part of it, then

$$A \times B = C_L + C_H\, 2^b = C_L + (-C_H) \bmod F_t$$

Thus, all one has to do is subtract the higher order b-bit register from the lower order b-bit register. The subtraction needs to be done according to (iii).

Example: Let $F_t = 2^2 + 1 = 17$

$$15 \times 10 = 150 = 14 \bmod 17$$

Now

$$150_{10} = (\underset{C_H}{1001}\ \ \underset{C_L}{0110})_2, \text{ and by (i)}$$

$C_H = 1001 \rightarrow 0110$

$\quad\quad\quad\ \ +10$

$\quad\quad\quad\ 1000$

$C_L = 0110$

$(-C_H) = \underline{1000}$

$\quad\quad\ 1110 = 14 \bmod 17$

(v) Multiplication by a power of 2

If x is taken as 2 or a power of 2 (in 3.7), the only multiplications involved in taking the Fermat transform are those by some power of +2. These multiplications are particularly simply to implement in arithmetic modulo F_t. Suppose one needs to multiply A by 2^K, $0 < K < b$, all one needs to do is shift to the left the content of the register by K bits, and subtract the K overflow bits (assuming double b-bit registers). If K is outside the range $0 < K < b$, one makes use of the fact $2^b = -1 \bmod F_t$.

Computation of the inverse transform requires multiplications by negative powers of 2 which can be converted to positive powers by the following relationship

$$2^{-K} = -2^b \cdot 2^{-K} = -2^{b-K} \bmod F_t \qquad (5.5)$$

Example: Let $F_t = 2^4 + 1 = 17$

(i) $15 \times 2^2 = 60 = 9 \bmod 17$ 15 = 0000 1111

Shift left 2 positions $\frac{0011}{C_H}\ \frac{1100}{C_L}$

$C_H = 0011 \rightarrow 1100$ $\quad C_L = 1100$

$+10$ $\quad (-C_H) = 1110$

$-C_H = 1110$ $\quad 11010$

-1

$1001 = 9 \bmod 17$

(ii) $13 \times 2^{-3} = 13 \times (-2^{4-3}) = 13 \times (-2) = -26 = 8 \bmod 17$

$$13 = \underset{C_H}{0000} \quad \underset{C_L}{1101}$$

Shift right circularly 3 positions $\underset{C_H}{1010} \quad \underset{C_L}{0001}$

$$C_L = 1010 \quad 0101 \qquad C_H = 0001$$
$$\underline{+10} \quad (-C_L) = \underline{0111}$$
$$-C_L = 0111 \qquad 1000 = 8 \bmod 17$$

For the implementation of the Fast Fermat Transform, unlike the FFT, one does not need to store the powers of x. For serial arithmetic, one could have a register which stores the shift amount K, and as one goes along the Fast Fermat Transform flow chart, one continually update the shift amount. This realization of b-bit arithmetic involves, as previously said, some input quantization error when $(-1 = 2^b)$ occurs as one input sample, as well as the extremely small, but realistic, probability of a complete data block in error when a (-1) occurs as the result of a FNT computation.

It is, of course, desirable to compute the FNT exactly. The difficulties in performing binary arithmetic modulo F_t, become apparent when one considers, for example, multiplication or addition in a ring of integers modulo F_t, involving the binary representation of $-1 = 2^b$. For example, when $b = 4$, $2^b+1 = 17$ the product of 10000 (-1) with itself is 00001(+1).

A technique for the exact computation of the FNT and its implementation are described by McClellan [26]. McClellan's approach involves the definition of a new binary code representation for the integers modulo F_t.

Given a binary representation of (b+1) bits,

$$A = [a_b, a_{b-1} \cdots a_o] \tag{5.6}$$

this new code is described as follows. If

$$a_b = 1 \quad \text{then } A = 0$$

$$a_b = 0 \quad \text{then } A = \sigma_{b-1}\, 2^{b-1} + \sigma_{b-2}\, 2^{b-2} + \ldots + \sigma_o \tag{5.7}$$

where

$$\sigma_j = \begin{cases} 1 & \text{if } a_j = 1 \\ -1 & \text{if } a_j = 0 \end{cases} \tag{5.7a}$$

Example: Let $F_t = 2^b+1 = 2^4+1 = 17$, $b = 4$

10000 represents zero

00111 $-2^3 + 2^2 + 2^1 + 1 = -1 = 16 \bmod 17$

01011 $2^3 - 2^2 + 2 + 1 = 7 \bmod 17$

and 10101 is an illegal combination, since the only allowed number with the most significant bit (MSB) equal to 1 is 10000 (zero). Consider arithmetic operations using this number representation.

(i) Multiplication by a power of two.

If the number is zero (i.e., $a_b = 1$) one does nothing. If the number is nonzero, the low order b bits are circularly shifted to the left a number of places equal to the power of 2, and a bit is replaced by its complement as it enters the least significant bit position (LSB).

Example: Let $F_t = 2^b+1 = 2^4+1 = 17$, $b = 4$

Using the proposed code

$$01100 = 2^3 + 2^2 - 2 - 1 = 9 \bmod 17$$

applying the above rule

$$9 \times 2 = ?$$

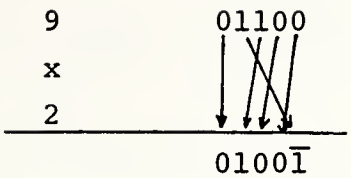
4 lower order bits

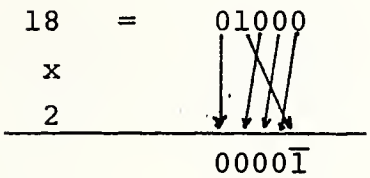
$$\begin{array}{ll} 9 & 01100 \\ \times & \\ 2 & \\ \hline & 0100\bar{1} \end{array}$$

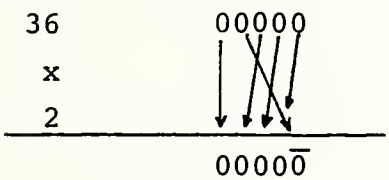
$$18 = 1 \bmod 17 = 01000 = 2^3-2^2-2-1 = 1 \bmod 17$$

Further,

$9 \times 8 = ?$







$$72 = 4 \bmod 17 = 00001 = -2^3-2^2-2+1 = -13+17 = 4 \bmod 17.$$

In a hardware implementation the MSB is used as a control bit. If it is one then the number is zero and the rotation is inhibited. This is characteristic of all operations using this new coding scheme.

(ii) Negative of a number

This is done by complementing the low order b bits except in the case where $a_b = 1$. Again the MSB is a control bit that inhibits the operation if it is one.

Example: Let $F_t = 2^b+1 = 2^4+1 = 17$; $b = 4$

Using the proposed code

$$A = 01101 = 2^3 + 2^2 - 2 + 1 = 11 \bmod 17$$

and by applying the above rule

$$-A = 00010 = -2^3-2^2+2-1 = -11 \bmod 17$$

(iii) Addition

If either or both of the operands for addition are zero (i.e., $a_b = 1$ or $c_b = 1$), then there is no addition to take place. That is, these special cases can be sensed and the addition inhibited. Now consider the addition of two numbers A and C where $A \neq 0$ and $C \neq 0$.

Let

$$A = a_b\, a_{b-1} \cdots a_o \quad \text{with} \quad a_b = 0 \tag{5.8}$$

and

$$C = c_b\, c_{b-1} \cdots c_o \quad \text{with} \quad c_b = 0$$

Interpret the b LSB's of A and C as the binary representation of $\hat{A}$ and $\hat{C}$, and form the same $\hat{A}$ and $\hat{C}$ using unsigned binary addition to obtain $\hat{D}$.

That is

$$\hat{A} = a_{b-1}\,2^{b-1} + a_{b-2}\,2^{b-2} + \ldots + a_o$$

$$+ \qquad (5.9)$$

$$\hat{C} = c_{b-1}\,2^{b-1} + c_{b-2}\,2^{b-2} + \ldots + c_o$$

$$\hat{A}+\hat{C} = \hat{D} = d_b\,2^b + d_{b-1}\,2^{b-1} + d_{b-2}\,2^{b-2} + \ldots + d_o$$

It is possible to deduce from $\hat{D}$, the desired sum

$$D = (A+C) \bmod 2^b+I \ [26].$$

Notice that

$$A = 2\hat{A} + 2 \ [26] \qquad (5.10)$$

Example: Let $F_t = 2^b+1 = 2^4+1 = 17$

Using the proposed code

$$A = 01010 = 2^3 - 2^2 + 2 - 1 = 5 \bmod 17$$

by (5.9)

$$\hat{A} = (1)\ 2^3 + (0)\ 2^2 + (1)\ 2^1 + 0\ (2^0)$$

$$= 8 + 2 = 10$$

and,

$$\begin{array}{rl} & 2\hat{A} = 2(10) = 20 \\ & + \\ & \quad 2 \\ \hline 2\hat{A}+2 = & 20 + 2 = 22 = 5 \bmod 17 = A. \end{array}$$

To deduce from $\hat{D}$ the desired sum D, verify that

$$D = A+C = 2\hat{A}+2 + 2\hat{C}+2 = 2\hat{A}+2\hat{C}+4 \bmod 2^b+1 .$$

Example: Let $F_t = 17$

and

$A = 01010 = 5 \quad \hat{A} = (1)2^3 + (0)2^2 + (1)2 = 8+2 = 10 \bmod 17$

$C = 00011 = 8 \quad \hat{C} = (1)2^1 + (1)2^0 = 3 \bmod 17$

So

$$D = 2\hat{A} + 2\hat{C} + 4 = 2(10) + 2(3) + 4 = 30 = 13 \bmod 17$$

and, checking

$$D = \hat{C} + A = 8 + 5 = 13 \bmod 17.$$

If D can be expressed as

$$D = 2\bar{D} + 2 \bmod 2^b + 1 \qquad (5.12)$$

with $\bar{D}$ a b-bit number, then the b-bits of $\bar{D}$ are the b-LSB's of D [26].

Example: Let $F_t = 17$

and

$$D = 01010 = 2^3 - 2^2 + 2^1 - 1 = 10 - 5 = 5 \bmod 17$$

Since D can be expressed as:

$$D = 2\bar{D} + 2 \bmod 17$$

where

$$\bar{D} = 1010 = (1)2^3 + (0)2^2 + (1)2 + (0)1 = 8 + 2 = 10$$

(check: $D = 2\bar{D} + 2 = 2(10) + 2 = 22 = 5 \bmod 17$)

the 4 bits of $\bar{D} = 1010$ are the 4 LSB's of D = 01010.

There are two cases depending on the value of d_b in (5.9).

1 - If $d_b = 1$, then by [26]

$$\hat{D} = 2^b + D' = D' - 1 \bmod 2^b+1 \qquad (5.13)$$

Example: Let $F_t = 17$

and

$$A = 01010 = 5 \bmod 17$$

$$C = 01011 = 7 \bmod 17$$

one has

$$\hat{A} = (1)2^3 + (0)2^2 + (1)2^1 + (0)1 = 8 + 2 = 10$$

$$\hat{C} = (1)2^3 + (0)2^2 + (1)2^1 + (1)1 = 8 + 2 + 1 = 11$$

$$\hat{D} = (1)2^4 + (0)2^3 + (1)2^2 + (0)2^1 + 1 = 21 = 4 \bmod 17$$

Comparing with

$$\hat{D} = d_b\, 2^b + d_{b-1}\, 2^{b-1} + \ldots + d_o$$

one verifies that

$$d_b = 1$$

in these conditions

$$\hat{D} = 2^4 + D' \bmod 17$$

where, by (5.13)

$$D' = \hat{D} + 1 = 4 + 1 = 5.$$

Then

$$\hat{D} = 16 + 5 = 21 = 4 = 5-1 = 4 \bmod 17,$$

checking (5.13). Thus,

$$D = (2\hat{A} + 2\hat{C} + 4) \bmod 2^b+1 = (2\hat{D}+4) \bmod 2^b+1 \qquad (5.14)$$

$$D = (2C' + 2) \bmod 2^b + 1$$

and

$$\overline{C} = C' \qquad (5.14a)$$

Example: Let $F_t = 17$

$$\begin{array}{rl} A = & 01010 = 5 \\ \underline{C = } & \underline{01011 = 7} \\ D = A+C = & \quad = 12 \end{array}$$

one has

$$\begin{array}{rl} \hat{A} = & 10 \\ \underline{\hat{C} = } & \underline{11} \\ \hat{D} = \hat{A}+\hat{C} = & 21 \end{array}$$

so

(1) $D = 2(\hat{A} + \hat{C}) + 4 \bmod 2^b + 1$

$D = 2(10+11) + 4 = 42 + 4 = 46 = 12 \bmod 17$

(2) $D = (2\hat{D} + 4) \bmod 17$

$D = 2(21) + 4 = 46 = 12 \bmod 17$

(3) In the previous example $D' = 5$

so

$D = 2(5) + 2 = 12 \bmod 17$

The condition $\overline{D} = D'$, in (5.14a), can be verified:

$A = 01010 = 5$

$+$

$C = 01011 = 7$

$D = A+C = 10101$

$\quad\quad\quad

2 - If $d_b = 0$ in (5.9), then [26]

$$D = 2D' + 4 \bmod 2^b+1 \qquad (5.15)$$

and

$$\overline{D} = D' + 1 \qquad (5.16)$$

Example: Let $F_t = 17$, $b = 4$

$$A = 00\ 111 = 16 \bmod 17$$

$$C = 01\ 011 = 7 \bmod 17$$

$$10\ 010$$

$$0$$

$$D = 00\ 010 = -8-4+2-1 = -11 = 6 \bmod 17$$

Since by (5.12)

$$\overline{D} = (0)2^3 + (0)2^2 + (1)2 + (0) = 2$$

and by (5.16)

$$D' = 1$$

checking,

$$D = 2\ D' + 4 \bmod 2^b+1 \quad \text{by (5.15)}$$

$$D = 2(1) + 4 = 6 \bmod 17.$$

Notice, that, this way of performing addition results in an extra level of add, as in the case of 1's complement arithmetic.

In 1's complement arithmetic, the output carry is added to the LSB.

In this new mod(2^b+1) arithmetic, one takes the output carry, complements it and adds it to the LSB. That is, this new arithmetic is only as complex as the 1's complement arithmetic.

There is a small amount of additional complexity due to the control bit, but this acts only as an inhibit signal.

Example:

(1) 01111 = +8+4+2+1 = 15 mod 17

00011 = -8-4+2+1 = -9 = 8 mod 17

(1)0010

0

00010 = -8-4+2-1 = -11 = 6 mod 17

(2) 01111 = 15 mod 17

10000 = 0

01111 = 15 mod 17

MSB = 1 inhibits the addition

(3) 01011 = 8-4+2+1 = 7 mod 17

(+)00100 = -8+4-2-1 = -7 mod 17

(0)1111

1

10000 = 0 mod 17

In this last example note that the second add automatically produced the control bit indicator, for the special case of zero.

How does one convert from a binary coded representation of numbers to this new representation?

The code conversion between a binary representation and this new code falls into two cases.

Let M be a number which is represented in both codes. Let

$$m_b \; m_{b-1} \cdots m_o \tag{5.16}$$

be the binary representation of M.
Let

$$\bar{m}_b \; \bar{m}_{b-1} \cdots \bar{m}_o \tag{5.17}$$

be the new representation.

Also, let $\hat{M}$ be the number represented by interpreting (5.17) as a binary code. The conversion rules are as follows:

1) If $M = 0$, then $\bar{m}_b = 1$, $\hat{M} = 0$, and $m_K = 0$ for $K = 0, 1, \ldots b-1$.
   This is a special case and is done separately.

2) If $M \neq 0$, then $\bar{m}_b = 0$ and

$$M = (2\hat{M} + 2) \bmod 2^b + 1 . \tag{5.18}$$

Conversion from the new code to binary is implemented by forming $2\hat{M} + 2$ and comparing this sum to 2^b.

If the sum is larger than 2^b, then 2^b+1 is subtracted to give the proper binary representation of M.

Example: Let $F_t = 17$

$M = 01011$ new code = $2^3-2^2+2+1 = 7 \bmod 17$

$\hat{M} = 8 + 2 + 1 = 11$

By the above rule

$$2\hat{M} + 2 = 2(11) + 2 = 24 = 7 \bmod 17$$

and

$7 = 00111$ binary representation of M.

If the binary representation is given the sum

$$M + 2^b - 1 \qquad (5.19)$$

is formed. If the result is odd, 2^b+1 is subtracted; and finally this result is right shifted one place. The resulting b bits are the b LSB's

$$\bar{m}_{b-1}\ \bar{m}_{b-2} \cdots \bar{m}_o \cdot$$

Example: Let $F_t = 17$, $b = 4$.

Given the binary representation of M = 11111 = 14 mod 17 to obtain new code:

(i) form

$$M + 2^b - 1 = 14 + 16 - 1 = 29$$

the result is odd then by the above rule.

(ii)

$$29 - 17 = 12$$

(iii)

$$\frac{12}{2} = 6 \qquad 00110 \text{ binary representation}$$

(iv) new code 4 LSB's of the binary obtained in (iii), i.e.,

new code 00110 = -8+4+2-1 = -3 = 14 mod 17

Notice that the code described so far, due to McClellan [26], is a special case of a more general class of code translations discussed by Leibowitz [27].

These translations involve the one-to-one representation of a number A in the ring of integers modulo F_t, as the binary number corresponding to

$$R\,A - 1 \bmod F_t \tag{5.20}$$

where R is any integer in the ring with an inverse [27].

Notice that the code representation of McClellan [26] corresponds to the case of [27]

$$R = 2^{b-1} + 1 \bmod F_t \qquad (5.21)$$

Example: Let $F_t = 2^b + 1 = 2^4 + 1$, $b = 4$

then

$$R = 2^{4-1} + 1 = 2^3 + 1 = 9 \bmod F_t$$

Using (5.20), for A = 5 mod 17 the code will be given by

$$RA-1 = 9(5) - 1 = 44 = 10 \bmod 17$$

$$10_{10} = 01010 \quad \text{code}$$

checking, using (5.7)

$$01010 = +8 - 4 + 2 - 1 = 10 - 5 = 5 \bmod 17.$$

Notice that the simplest code translation corresponds to R = 1, for any value of b. Leibowitz [27] concentrates on the resulting binary arithmetic for the code translation corresponding to R = 1.

Recall that to represent all integers in the ring modulo F_t requires (b+1) bits. The additional bit is required in order to represent the number $2^b = -1 \bmod F_t$.

In order to overcome the problem of performing binary arithmetic with this additional bit, one allows the additional bit to be a 1 only when the number to be represented is a zero.

One way to do this is achieved by subtracting 1 from the normal binary representation of every integer in the ring and corresponds to the above set of code translations with R = 1.

Example: Let $F_t = 2^b + 1 = 2^4 + 1 = 17$, $b = 4$

Normal value	Binary representation	Diminished -1 value (R = 1)
0	00000	1
1	00001	2
2	00010	3
3	00011	4
4	00100	5
5	00101	6
6	00110	7
7	00111	8
8	01000	9(-8)
9(-8)	01001	10(-7)
10(-7)	01010	11(-6)
11(-6)	01011	12(-5)
12(-5)	01100	13(-4)
13(-4)	01101	14(-3)
14(-3)	01110	15(-2)
15(-2)	01111	16(-1)
16(-1)	10000	0

(5.22)

In the diminished -1 number representation the b-least significant bits (LSB's) indicate the normal value of the number. The numbers from 1 to 2^b are represented in order by the binary numbers from 0 to $2^b - 1$.

Using this representation, the arithmetic operations necessary to perform convolution by FNT's, will be discussed next.

1. Negation

It can be seen from (5.22) that each of the negative numbers ($>2^{b-1} = 8$) is the b-LSB's complement of its positive counterpart.

Example: Let $F_t = 17$

Diminished -1 value	Binary representation
A = 6	00101
-A = -6 = 11	01010

Notice that if the MSB is 1, the negation is inhibited. Thus, the negative of a nonzero number in the diminished -1 representation is the complement of its b-LSB's.

2. Addition

To perform addition of two numbers represented as (A-1) and (B-1)

$$(A-1) + (B-1) = (A+B-1) -1$$

and thus

$$(A+B-1) = [(A-1) + (B-1)] + 1 \qquad (5.23)$$

Since the $(b+1)^{th}$ bit of the addends is used only to inhibit addition if an addend is zero, addition of nonzero addends involves only the b-LSB's.

Equation (5.23) indicates that a 1 must be added to the sum of two diminished (-1) numbers to provide a correct result. When a carry is generated from the b-bit sum, a residue reduction modulo F_1 requires the subtraction of a 1 since $2^b = -1 \bmod F_t$ and no corrective addition is necessary. Thus to add to numbers in diminished -1 representation:

1) If the MSB of either addend is 1, inhibit the addition and the remaining addend is the sum.
2) If the MSB of both addends are 0, ignoring the MSB, add the b-LSB's, complement the carry and add it to the b-LSB's of the sum.

The $(b+1)^{th}$ bit or MSB of the resulting sum is the carry out of the b^{th} bit.

Example: Let $F_t = 17$, b = 4

1.	Diminished -1 value	Binary representation
add	3	00010
+	2	00001
	5	00011
		1
		00100

2.

add	7	00110
+		
	0	10000
	7	00110

3.

add	13	01100
+		
	9	01000
		10100
		0
	22 = 5 mod 17	00100

4.

add	11	01010
+		
	6	00101
		01111
		1
	17 = 0 mod 17	10000

3. Code Translation

Let B represent the binary representation of a given number and D its diminished -1 representation.

To perform code translation from binary to diminished -1 representation, one does a diminished -1 addition of B and the binary representation of $2^b - 1$ [27].

Example: Let $F_t = 17$, $b = 4$.

Binary number	B	= 00101	= 5
	2^b-1	= 01111	
		10100	
		0	
Diminished -1 value	D	= 00100	

i.e., the binary number B = 00101 = 5 is represented in diminished -1 representation by

$$D = 00100 = (4) = (5-1).$$

The translation from diminished -1 representation to binary representation, is performed by complementing the MSB of D and adding it to the b-LSB's [27].

Example:

Diminished -1 D = 00010 3

1

Binary B = 00011

Example: D = 10000 0 Diminished -1 representation

0

B = 00000 Binary representation.

4. Subtraction

One can perform subtraction in the dimished -1 arithmetic by negating the subtrahend and adding it to the minuend.

Example:

	Diminished -1	Binary representation
subtract	8	00111
-	6	01010
		10000
		0
	2	00001

5. Multiplication by Powers of 2

In performing a multiplication if the multiplier or multiplicand are 0, as detected by the presence of a 1 in the (b+1)th bit, the multiplication is inhibited and the product is zero.

To perform multiplication of diminished -1 numbers by powers of 2, notice that:

$$(A-1)2 = (2A-1) - 1$$

and thus

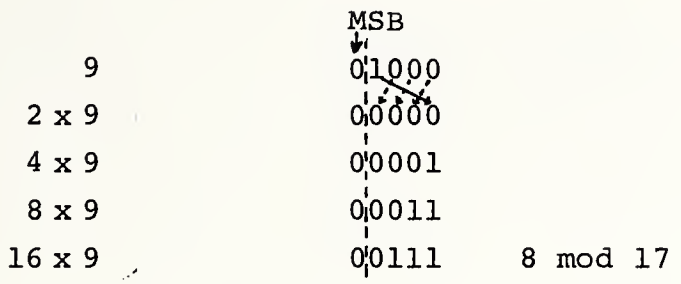
$$(2A-1) = (A-1)2 + 1 \qquad (5.24)$$

therefore, each multiplication by two involves a left shift, ignoring the MSB, and a corrective addition of a 1. If the bit shifted out from the b^{th} position is a zero, it is complemented and shifted into the LSB in order to accomplish the addition of a 1. If this bit is a 1, a subtraction of 1 is also required to accomplish a residue reduction ($2^b = -1$). With the corrective addition of +1, these cancel out and a 0 is shifted into the LSB.

Thus for each factor of 2, a left circular shift of the LSB's is required and the bit circulated into the LSB is complemented.

Example: Let $F_t = 17$, $b = 4$

$$16 \times 9 = 2^4 \times 9 = 2 \times 2 \times 2 \times 2 \times 9 = 8 \bmod 17$$



6. General Multiplication

The last operation required to carry out convolution with the FNT is a general multiplication by any two integers modulo F_t.

To perform a multiplication of the numbers A and B represented as (A-1) and (B-1) in diminished -1 number representation system, notice that

$$(A-1)\ (B-1) = A \cdot B - (A+B) + 1$$

$$= (A \cdot B-1) - (A+B-1) + 1$$

and the desired result is

$$(A \cdot B-1) = (A-1)(B-1) + (A+B-1) - 1 \qquad (5.25)$$

thus, to carry out such an operation, ignore the MSB's, perform a binary multiplication of the diminished -1

representation of A and B, add this result to the b-LSB's of the diminished -1 addition of A and B and then perform a residue reduction by a diminished -1 subtraction of the b-MSB's from the b-LSB's.

Notice that this particular general multiplication scheme is not applicable with code translations other than that corresponding to R = ± 1. As discussed previously, if the MSB of either the multiplier or the multiplicand is 1, then the multiplication is inhibited and the result is set to zero.

Example: Let $F_t = 17$, b = 4.

Multiply

```
    13
 x
    11
   143   =  7 mod 17
```

binary multiply

```
   01100
   01010
  011000
 0110000
 01111000
   00110
 01111110
```

binary add

diminished -1 add

```
   01100
   01010
  (1)0110
      0
   00110
```

residue reduction

```
    1000
  (1)0110
      0
   00110    7 mod 17.
```

An alternate multiplication technique can also be considered. Assuming that the numbers are determined to be nonzero and a multiplication is required, a translation to normal binary coding is performed. Following a binary multiplication, a residue reduction by diminished -1 subtraction of the b-MS's of the product from the b-LSB's is carried out. The result is the desired product.

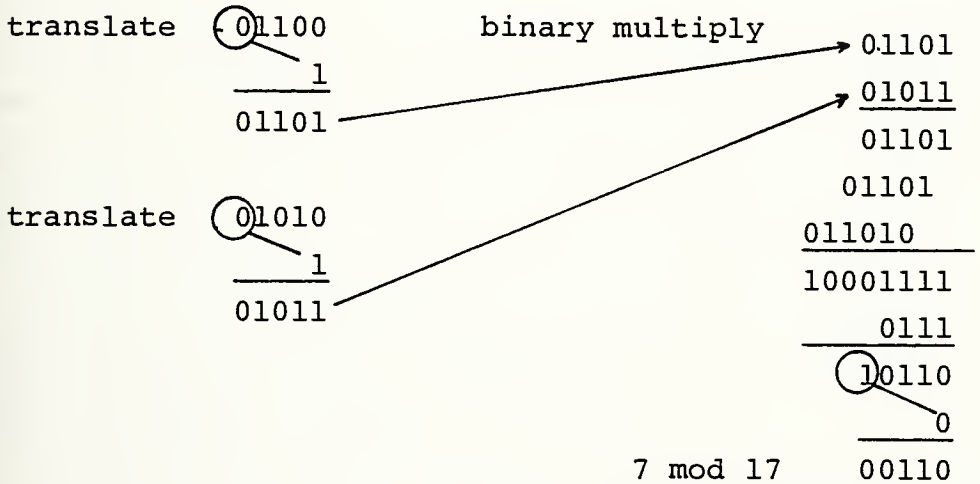
Example: Let $F_t = 17$, b = 4

multiply 13

x

11

143 = 7 mod 17



Since in most cases the translation from diminished -1 to binary will be simpler and faster then a general binary or diminished -1 addition, the second technique is the most preferable.

To conclude part A of this section, one can say that Leibowitz [27] presents a binary arithmetic applicable to

the exact computation of the FNT, that this diminished -1 representation is mathematically simpler than that of McClellan [26]. Also the hardware presented in [26] to compute the FNT, and discussed in part B of this section, can be applicable to this new technique with the exception of the code translation.

B. SOFTWARE AND HARDWARE REALIZATIONS OF THE FNT

The Fermat Number Transform has been proposed as an aid to the rigid and precise computation of convolution for digital filtering applications. Since this transform does not require multiplications, it is considerably faster than the FFT [17].

In what follows, a discussion of a software implementation of the FNT, due to Agarwal and Burrus [4], as well as the description of a special purpose hardware to compute the FNT, realized by McClellan [26], is presented.

In their assembly language realization of the FNT, on the IBM 370/155 computer, Agarwal and Burrus define a binary arithmetic modulo $F_t = 2^b+1$, $b = 2^t$.

Notice that the IBM 360/370 series uses a 32-bit word length for fixed-point arithmetic, and therefore is well suited for the implementation of convolution using the FNT modulo $F_5 = 2^{32} + 1$.

It has two's complement representation of negative integers, i.e.,

$$(-x) \text{ is represented as } 2^{32} - x \qquad (5.26)$$

Since in arithmetic mod F_t one wants negative integers to be represented as a complement of $2^{32} + 1$, i.e.

$$(-x) \text{ is to be represented as } (2^{32}+1-x) \qquad (5.27)$$

One adds 1 whenever a negative number is encountered in data.

As noted before with b bits, the quantity $2^b = -1 \bmod F_t$ has no representation. Thus in the 370/155, one cannot represent $2^{32} = -1$. If a (-1) is encountered in the data it is rounded either to 0 or to -2. If data is uncorrelated, the probability that (-1) will appear after an arithmetic operation during the various stages of the FNT computation is roughly

$$2^{-32} \simeq 10^{-10} \qquad (5.28)$$

To add two 32-bit integers modulo F_5, the logical add instruction (ALR) is used which adds the two integers and sets the condition code depending on carry or no carry. After the logical add instruction, a conditional branch is taken depending on the condition code. If the condition code indicates a carry bit in the logical add operation, one is subtracted from the result, otherwise it is left unchanged. This sequence of operations completes one addition modulo F_5. Similarly, subtraction modulo F_5 is performed using a logical subtraction (SLR) instruction followed by a conditional

branch instruction. To multiply a number by

$$2^K \bmod F_5 , \quad 0 < K < 32$$

the number is loaded in the odd register of an even-odd fair of registers. The even register is filled with zeros and this double register is shifted left by K positions using a shift left double logical (SLDL) instruction. After the shift operation, the even register is subtracted from the odd register, modulo F_5. This sequence of instructions completes multiplication by $2^K \bmod F_5$.

To multiply a number by

$$2^{-K} \bmod F_5 , \quad 0 < K < 32 ,$$

the number is loaded in the even register of an even-odd pair of registers, the odd register is filled with zeros and this double register is shifted right by K positions using a shift right double logical (SRDL) instruction. After the shift operation, the odd register is subtracted from the even register, modulo F_5. This sequence of operations completes multiplication by $2^{-K} \bmod F_5$. To multiply two numbers mod F_5 requires a somewhat larger number of operations. First one determins the signs of the two integers and multiply the signs. If any of the numbers is detected as negative, its absolute value mod F_5

is taken by taking its two's complement and adding one to it. Their absolute values are multiplied (MR) to obtain a 64-bit double-register product, then the even register (higher order 32 bits) is subtracted from the odd register (lower order 32 bits) mod F_5. Finally, if the product of signs was detected negative, one multiplies the result by (-1) by taking its two's complement and adding one to it [5.27].

Assembler subprograms were written [4] to compute Fast Fermat Number Transforms and inverse Fermat number transforms for an sequence length from 2^1 to 2^6, taking x as a power of 2. A decimation in frequency algorithm with normally ordered input and bit reversed output was used for the computation of the Fast Fermat Number Transform [4]. A decimation in time algorithm with bit reversed input and normally ordered output was used for the computation of the fast inverse FNT.

For both of these subprograms, the only multiplications required were by a power of 2, which were implemented as discussed above. Two more subprograms were written to compute fast FNT's and inverse FNT's for length - 128 sequences, using $\alpha = \sqrt{2}$ given by (4.16).

Multiplications required to multiply the two transforms are general multiplications mod F_5 and are performed as discussed earlier.

To cyclically convolve sequences longer than 128, two-dimensional convolution schemes were used.

One such scheme was 2 by 128 convolutions for length 256 sequences discussed in section 10 of [18]. Another program was written to convolve long one-dimensional sequences using two-dimensional FNT, as discussed in Appendix B. In Section 6, results of the comparison with the FFT are presented.

The special attributes of the FNT led several researchers to consider seriously special-purpose hardware implementations of the transform. The machine constructed by McClellan [26], is an implementation of a 64-point, 16-bit FNT (modulo $F_4 = 2^{16} + 1$). This hardware system applies, the second coding scheme, described in part A of this section, to the logic design of the butterfly of the FNT algorithm. The fast FNT algorithm implemented is a radix-2 constant geometry decimation in frequency (DIF) decomposition of the FNT.

Fig. 1 shows a flow diagram of this algorithm for a 16-point transform [38]. Although the constant geometry structure does not allow in place calculation of the transform, it simplifies the memory addressing because the addressing does not vary from stage to stage. The price one pays for this simplification is a doubling of the memory size required for the transform. However, 128 words per chip is a convenient level of integration with ECL (emiter coupled logic), thus making the constant geometry structure attractive [26].

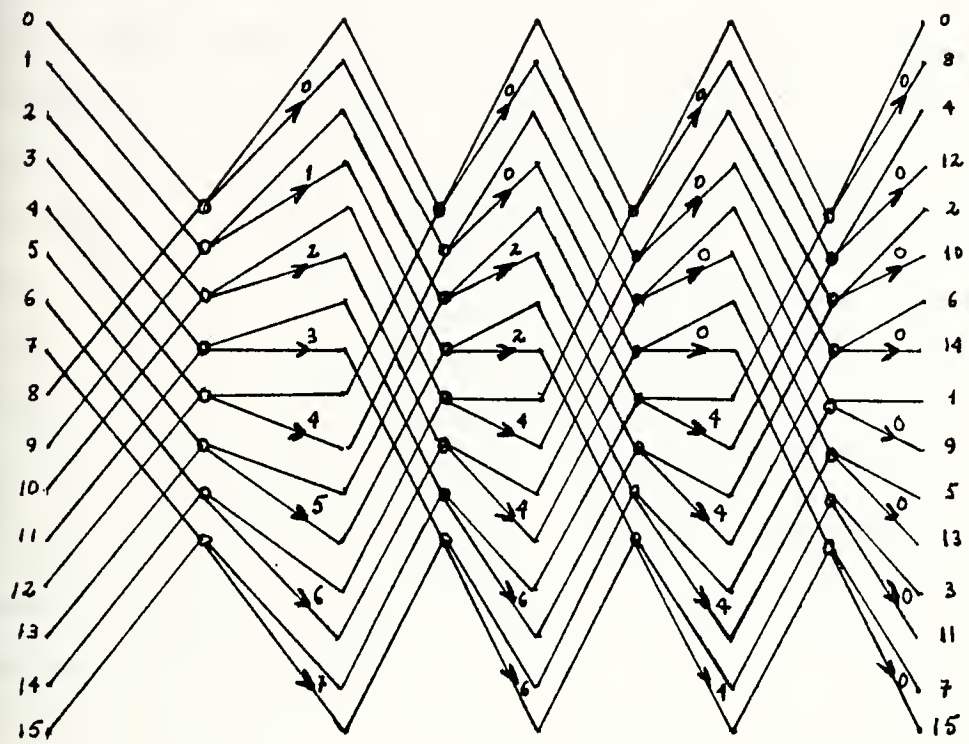


FIGURE 1. Flow Diagram of a Radix-2, 16 Point, Constant Geometry FFT Algorithm Using the Decimation in Frequency Structure

Fig. 2 is a block diagram of the complete system showing the four major subsystems. The computational element (CE) is a radix-2 DIF butterfly for the fast FNT algorithm; the memory element contains 128 x 17-bit words for use as intermediate storage during the computation of the transform; the control element is a hardwired implementation of the fast FNT algorithm [26]; and the input/output (I/O) section provides the interface with the fast digital processor (FDP). McClellan states that the goal in building this hardware was to construct a CE that would operate at a clock rate of 40 MHz. In order to achieve this speed, ECL 10K circuits were used. The basic gate in this logic family as a propagation delay of 2 ns, and thus these circuits are well-suited for very high-speed systems. Even with such high speed logic circuits, two levels of reclock and fast carry addition were used in the CE to realize a working system that runs reliably at 38 MHz [26].

Fig. 3 shows a functional diagram of the butterfly which consists of an adder, a subtracter, a rotator, input buffer registers R_A and R_B, reclock registers R_W, R_X and R_Y and an output register R_Z. Register transfers are made at each clock pulse, so that data are always flowing through the CE as would be the case in a pipelined fast FNT.

As the timing diagram in Fig. 4 shows, the output of the butterfly is only written into memory from R_Z at t_4 and t_5. During the other clock epochs the contents of R_Z may be changing but this does not affect the algorithm.

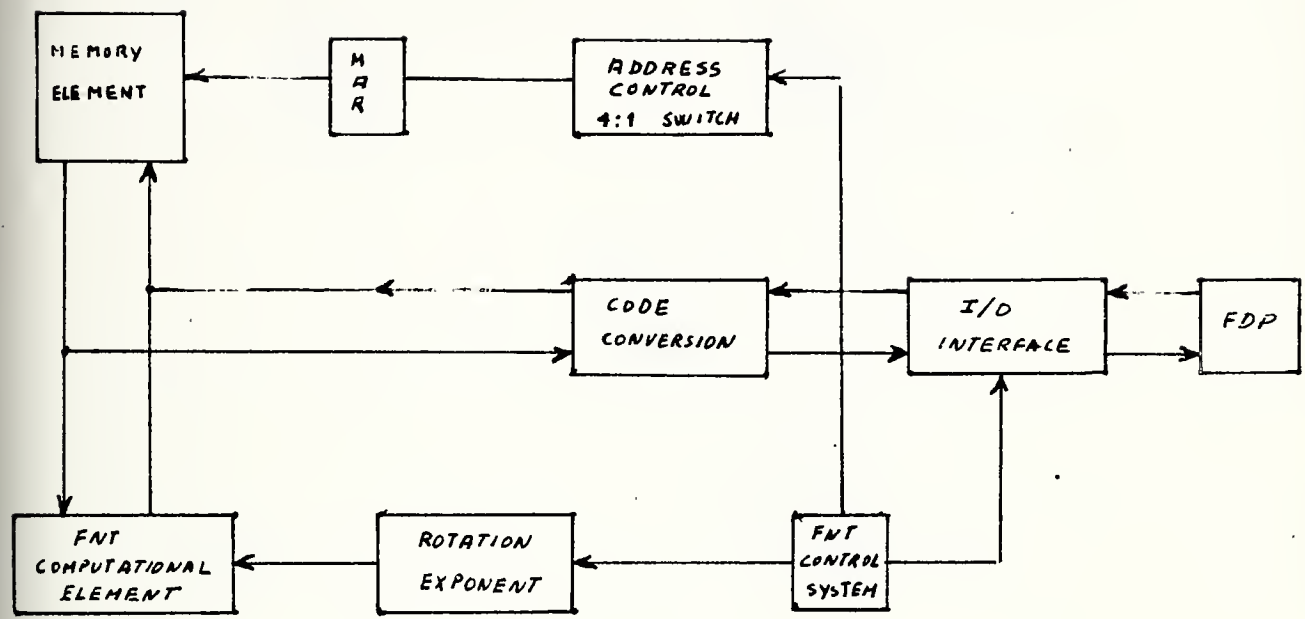


FIGURE 2. Block Diagram of the 64-Point FNT Hardware System Illustrating the Data Flow Between the Major Subsystems

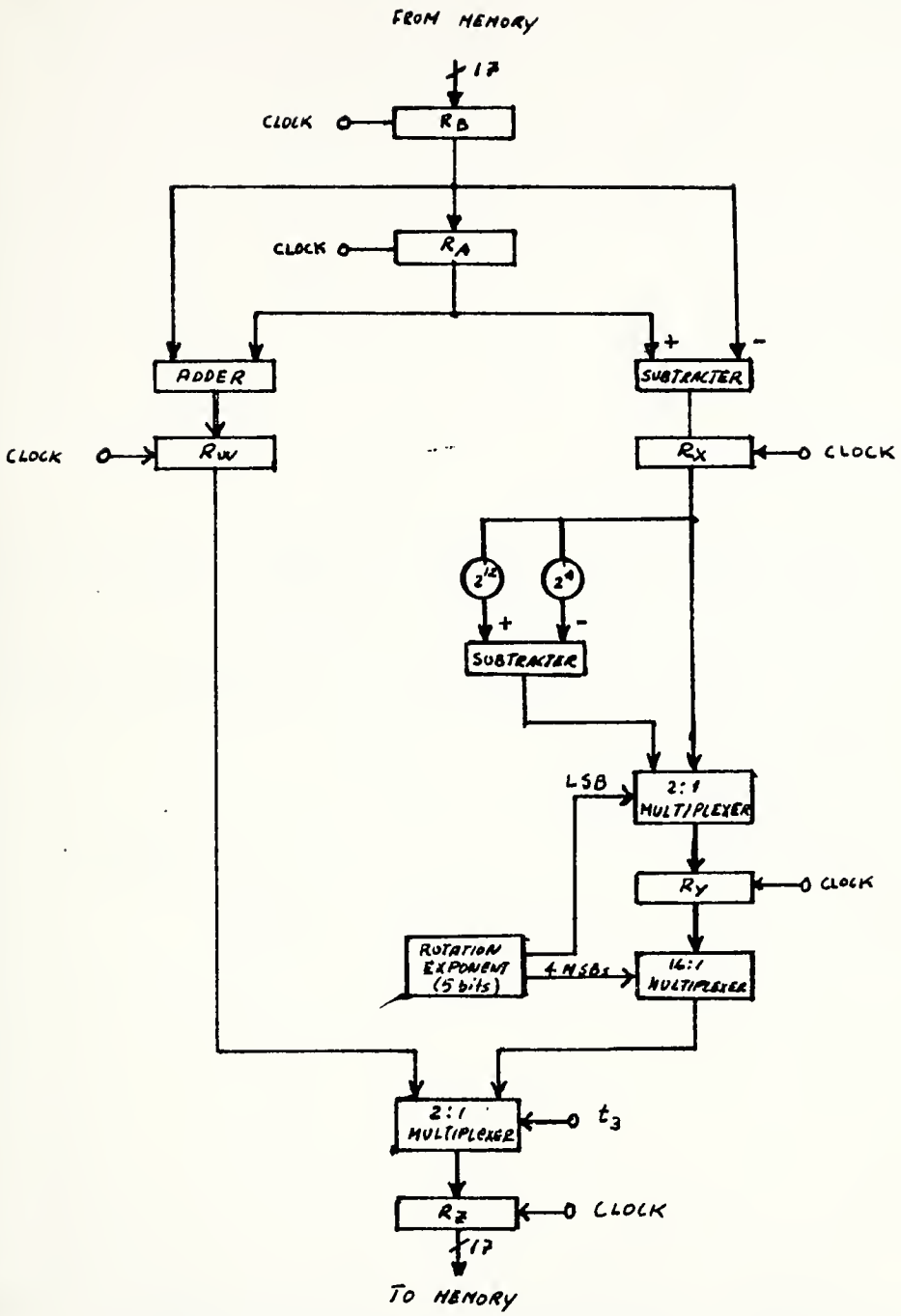


FIGURE 3. Register Transfer Diagram of the CE of the FNT Algorithm. Path on the Left Implements A+B and the Path on the Right (A-B) $\overline{2^K}$

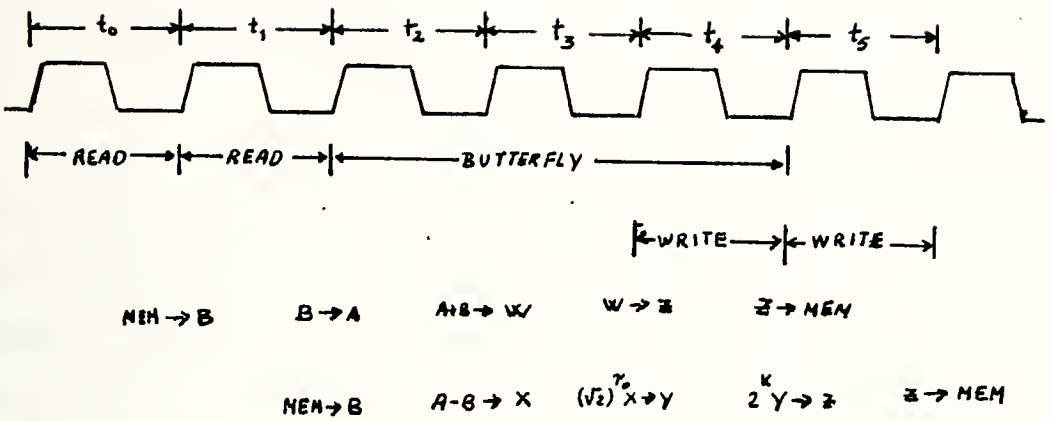


FIGURE 4. Timing Diagram for the Internal Clocking Operation of the FNT Butterfly. Register Transfers at Each Clock Pulse are Also Shown

Recall that in McClellan's code, the rule for addition of two nonzero numbers,

$$A = [a_{16}\ a_{15} \cdots a_o]$$

and

$$B = [b_{16}\ b_{15} \cdots b_o]$$ is as follows:

STEP1: Add the 16 LSB's of A and B with the carry in equal to zero

STEP2: Complement the carry out from step 1 and add it to the sum of step 1.

If either A or B is zero (i.e., b_{16} or $a_{16} = 0$), then the carry must be inhibited. Finally if both A and B are zero the MSB of the sum is set to one. Fig. 5 shows a realization of the addition process. The structure of Fig. 5 is inefficient in two respects. First of all, two 16-bit adders are required, although the second one is simple because one input is zero. Secondly the addition is very slow because the carry must propagate through the 16-bit adders. The use of carry look ahead (CLA) logic as in FIg. 6 will improve both situations. For details of the implementation using ECL building blocks see McClellan's paper [26].

Notice that the subtracter A-B, can be implemented by complementing B and adding it to A.

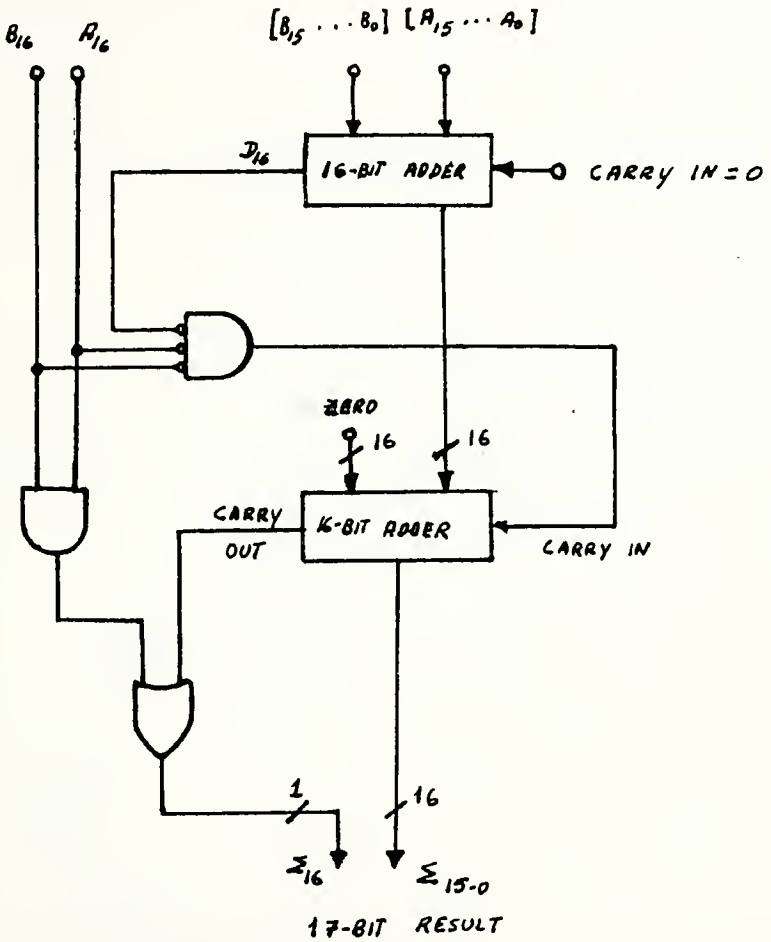


FIGURE 5. Implementation of Addition Modulo the Fermat Number $(2^{16}+1)$ Using Two 16-Bit Adders and the New Coding Scheme

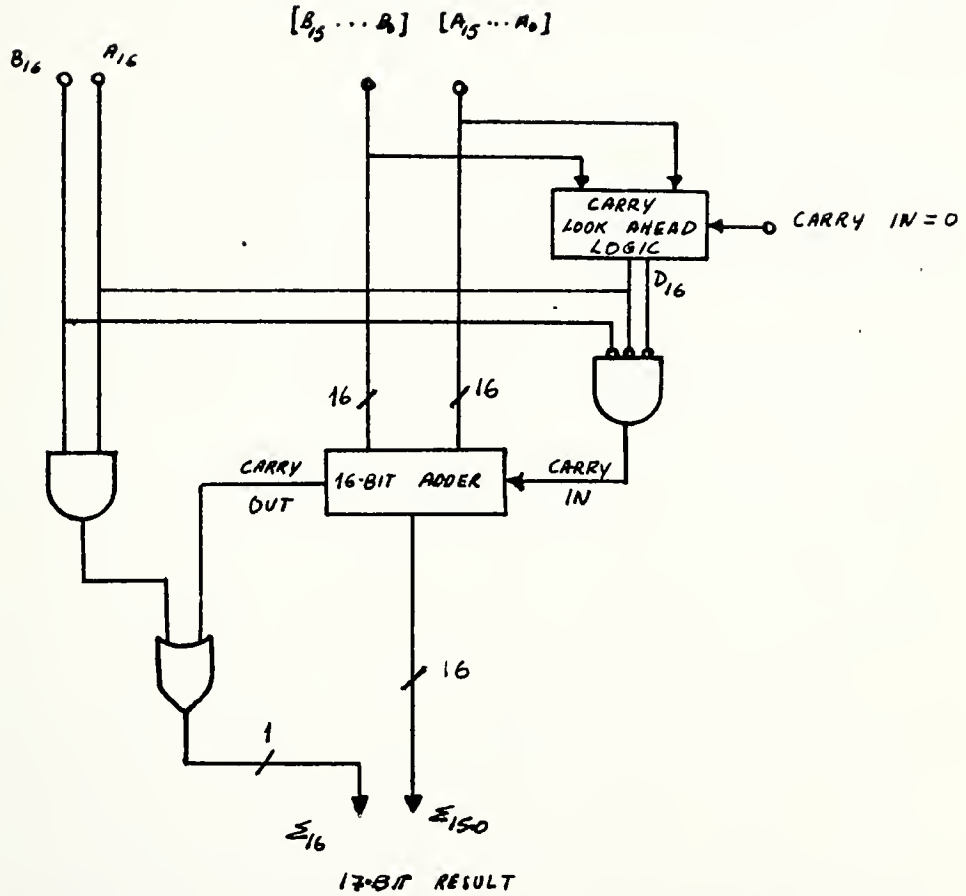


FIGURE 6. Improvement of the Fermat Number Adder by the Introduction of CLA Logic to Produce the End Around Carry Required in Fig. 5

The addition A+B completes the calculation of one output of the computational element (see Fig. 3). The result is held in the register R_W and then is moved to R_Z to be written back into memory. For the other output of the CE, the quantity A-B is stored in the reclock register R_x for subsequent rotation by a power of

$$\alpha = \sqrt{2} \text{ given by (4.16),}$$

$$\alpha = \sqrt{2} = 2^{b/4}(2^{b/2}-1) = 2^{16/4}(2^{16/2}-1) = 2^4(2^8-1) = 2^{12}-2^4 \tag{5.28}$$

The rotation by $\sqrt{2^K}$ is split into two stages.

In the first stage, the quantity X = A-B is multiplied by $\sqrt{2} = 2^{12} - 2^4$ if K is odd. The $\sqrt{2}$ multiplier is merely a subtracter (5.28).

A 2:1 multiplexer at the output of the subtracter selects whether the input is to be multiplied by $\sqrt{2}$ or by 1, and is controlled by the LSB of K [26]. The result of this calculation is stored in the reclock register R_y. The second stage of the rotation is a multiplication by a power of 2, namely [K/2]. ([ ] denotes the greatest integer function.) This multiplication is implemented as a 16:1 multiplexer controlled by the upper four bits of the binary representation of the power of 2. The shifting network is followed by a 2:1 multiplexer which selects which butterfly output (A+B or $2^K \cdot y$) is to be stored in

R_Z and then written back into memory. This multiplexer is controlled by t_3 and its output is

$$(2^K \cdot \underline{Y})\ F_3 + W \cdot t_3$$

where $\underline{Y}$ and W are the contents of R_Y and R_W respectively.

McClellan states that the logic for the computational element consists of 90 IC's, all of which are located on one board (18 x 16 in.).

In Section VI, a comparison between the complexity of various basic operations involved in computing Fermat Number Transforms vis-a-vis the FFT, in software as well as hardware implementations, will be presented. The software and hardware implementations discussed there will be those described in this section. The results will show only the efficiency of these implementations, not all the possibilities of NTT in general.

VI. FERMAT NUMBER TRANSFORM VERSUS FFT

The FNT provides an efficient and error-free means of computing cyclic convolutions. The purpose of this section is to compare this method with the standard implementation of convolution. Results obtained both by Agarwal and McClellan, respectively in software and hardware implementation of the FNT, are presented.

Computation of the FNT of length N requires on the order of $N \log_2 N$ additions, bit shifts and subtractions but no multiplications. The only multiplications required for our FNT implementation of cyclic convolution are the N multiplications required to multiply the transforms. This is a very efficient technique for computing convolution, but unfortunately, the maximum transform length for an FNT is proportional to the word length of the machine used. Agarwal and Burrus [17] showed that a very practical choice of a Fermat number for this application is $F_5 = 2^{32} + 1$, and that the FNT mod F_5 can be implemented on a 32-bit machine, namely the IBM 360/370 series computers.

Suppose one wants to calculate the convolution of two sequences x(n) and h(n) having b_1 and b_2 bit representations, respectively, and that the sequence length of both is N. Then the output y(n), given by (3.2), would have at the most [17] a

$$b_1 + b_2 + \log_2 N \qquad (6.1)$$

bit representation. To obtain the correct result b, the number of bits of the output, should be [17]

$$b \geq b_1 + b_2 + \log_2 N \tag{6.2}$$

Notice that a better bound on the output can be achieved [4].

Roughly speaking, one needs twice the number of bits to carry out the convolution using the FNT as compared to the fixed point FFT implementation of the convolution. But in the DFT, every data point is treated as a complex number [17] and therefore requires two words, one for the real part and one for the imaginary part. Thus in effect the hardware requirement for two transforms are about the same. Although for real data it is possible to make use of the symmetry properties of the DFT, they require extra computation and for the purpose of comparison it will be ignored, although Agarwal and Burrus had taken this into account for their IBM 370/155 implementation. In fact, they assumed, in the FFT implementation, each data point is represented by a b/2 bit real part and a b/2 bit imaginary part. One b/2 bit complex addition is equivalent [17] to two b/2 bit real additions, which are equivalent to a b-bit addition modulo F_t. Thus the complexity of addition/subtraction is the same in both the transforms. Similarly, it can be shown [4] that a b/2 bit complex multiplication is equivalent

to a b-bit multiplication modulo F_t. Computation of the FNT requires multiplications by a power of 2, which implemented as bit shifts and subtractions become much simpler operations compared to complex multiplications required in the FFT implementation.

To compute a length N fast FNT, N $\log_2 N$ additions/subtractions, and $(N/2) \log_2 (N/2)$ "multiplications" by some powers of 2 which are implemented as bit shifts and subtractions. To compute the convolution using the FFT, most of the time is taken in computing the complex multiplications required to compute the complex multiplications required to compute the transforms. A comparison with the FNT reveals that these complex multiplications are replaced by bit shifts and subtractions which are much faster operations. This results in considerable computational savings in the implementation of convolution.

The convolution required to multiply the two transforms is about the same for both the implementations.

To convolve long sequences using the two-dimensional FNT (Appendix B), the computational effort increases by, at most, a factor of 2 [4]. Still, the FNT implementation of convolution is much faster as compared to the FFT implementation.

Fermat number transforms have some additional advantages over the FFT. First, the FFT implementation requires storing all the powers of α (see 3.7) requiring a significant amount of storage which may be an important factor for a small

minicomputer or a special purpose hardware implementation. Second, fixed-point FFT implementation introduces a significant amount of round-off noise at the output, 6-8 bits depending on the data [39]. This degrades the signal-to-noise ratio during the filtering operations. The FNT implementation is error free, the only source of error is input A/D quantization.

Timings for various implementations of convolution and their comparison with the FFT implementation are shown by Agarwal and Burrus [4];

N	FFT msec	FNT msec	
32	16	3.3	
64	31	7.4	
128	60	16.6 *	
256	123	40.0 **	
256	123	80.0 ***	(6.3)
512	245	166.0 ***	
1024	530	340.0 ***	
2048	1260	720.0 ***	

* using $\alpha = \sqrt{2}$

** using 2 by 128 convolution

*** using the two-dimensional FNT

To compute these timings it is assumed that the transform of the h sequence has been precalculated. Thus timings are for computing x transforms, multiplications of the transforms, and their inverse transforms [4].

For cyclic convolution lengths up to 256, the FNT implementation of convolution have a factor of 3 to 5 speed advantage over the FFT implementation. It takes roughly 13-14 μsec to compute one butterfly in the computation of fast Fermat number transforms [4]. Timings for the computation of fast Fermat number transforms, are fairly well modeled by multiplying this time by the number of butterflies required in the computation. An assembler subprogram was written to compute one butterfly for the FFT algorithm [4]. The timing for this was 68 μsec, and this explains the difference in the timings of the FFT and FNT. Agarwal claims that since assembler subprograms were not optimized for time, it should be possible to further reduce their timings by 10 to 20 percent. Also, to compute one butterfly of the fast FNT's, three add/subtract logical instructions are required and since the arithmetic is done mod $2^{32}+1$, these three instructions are followed by three branch instructions (to take into account the carry bit). If the hardware was designed to do arithmetic mod $2^{32}+1$, these instructions could have been avoided resulting in a significant reduction in the computation time [4]. One objective of the construction of FNT convolver by McClellan was to evaluate the total system cost of an FNT convolver versus a pipeline FFT convolver, primarily in the speed regime applicable to radar signal processing.

The signal bandwidths encountered in radar signal processing (10-30 MHz) require a pipeline architecture for

either the FNT or FFT [40]. Typically, the overlap-save version of high-speed convolution is employed for real-time processing of the radar returns and a 50 percent overlap of the input data is common [26]. Furthermore, the length of the convolution to be implemented is assumed to be large (e.g., 512 or greater). Two cases will be considered: a length 1024 convolution of real data and a length 1024 convolution of complex data.

Four measures of hardware complexity are the basis of comparison [26]:

- the number of butterflies for output point,
- the number of reference spectrum multiplies for output point,
- the total amount of interstage delay line memory in the forward and inverse transforms,
- and the total amount of reference spectrum memory.

The FFT implementation will be considered first. For either real or complex data, it is assumed that the FFT implementation employs an 11-stage radix 2 pipeline FFT in both the forward and inverse transform. Notice that for real data, it is possible to do a length N transform with one length N/2 transform and some overhead to combine the real and imaginary parts [38]. However the overhead amounts to an additional butterfly so there is little, if anything to gain by using this fact in a pipeline FFT [26].

For the case of a length $2^{11} = 2048$ pipeline FFT convolver the number of butterflies per output point is

$$2 \log_2 N = 22 \tag{6.4}$$

assuming 50 percent convolution overlap. Likewise, two reference spectrum multiplies must be done per output point [26]. The amount of interstage delay line memory can be calculated from the formula [26]

$$IDM = \frac{N}{2}(r+1) \tag{6.5}$$

where r is the radix of the transform.

Thus, for two radix-2 pipelines, the total is

$$IDM = \frac{N}{2}(2+1)\cdot 2 = 3N = 6144 = 6K$$

words of memory. Finally, the reference spectrum requires 2K words of memory [26]. A pipeline FNT structure is identical to the pipeline FFT except in the butterfly where rotation by $e^{i2\pi K/N}$ (in the FFT case) is replaced by multiplication by $\sqrt{2^K}$.

Thus many of the results quoted above are applicable to the FNT. Since the FNT naturally processes real input data, the cases of real and complex convolution require different realizations. In both cases, however, a two-dimensional implementation of the convolution is required [31].

The length 1024 convolution of real signals can be implemented with 64 x 64 transforms [26].

McClellan claims that for the FNT convolver processing real data, the following results apply:

total number of butterflies = 9×2^{12} 36 butterflies per output point

total reference function multiplies = 2^{12} 4 multiplies per output point

total interstage delay memeory = 6·4 K

amount of reference function memory = 4K.

If the signal to be convolved is complex then one possible implementation is to handle the real and imaginary parts of data separately. The number of butterflies per output point, the interstage delay memory, and the reference spectrum are all doubled. However, the number of real multipliers for the reference function multiply is quadrupled because the multiplications are now complex [26]. These results can be summarized, assuming 1024 convolutions, by

	FFT Real or Complex Data	FNT Real Data	FNT Complex Data
Butterflies	22	36	72
Reference Multipliers	2	2	16
Interstage Delay Memory	6K complex words [a]	6·4K real words [b]	12·8 K real words
Reference Spectrum Memory	2K complex words	4K real words	8K real words

a One complex word will contain approximately 27 bits for typical high-precision radar applications.

b One real word contains 33 bits for FNT.

Notice that the FNT always requires more memory and more computational elements than the FFT. Hardware savings are possible because most of the hardware cost of the FFT is concentrated in the butterfly elements (up to 80 percent) [26] and because the FNT butterfly requires from one-third to one-sixth the hardware of an FFT butterfly. These remarks apply to the FNT where the data to be convolved are real, but where the data are complex, the situation becomes much worse because all measures of hardware complexity are increased by a factor of 2 or 4.

McClellan conclude that the FNT is a useful alternative to the FFT if the signal to be filtered is real and the computational elements are a major part of the overall system cost as in a pipeline architecture. Furthermore, for short length convolution (e.g., length 64) and for two-dimensional convolution the savings may be significant.

VII. CONCLUSIONS

It has been shown that, by working in a finite field or ring of integers modulo M, a large class of transforms exist that have the cyclic convolution property, i.e., the transform of cyclic convolution of two sequences is equal to the product of their transforms. These transforms are called Number Theoretic Transforms (NTT's) and they are a computationally efficient approach to performing the discrete convolution function. These NTT's are truly digital transforms, taking into account the quantization in amplitude and the finite precision of digital signals. They bear the same relation to digital signals as the DFT does to discrete-time or sample data signals and the Fourier or Laplace transforms do to continuous time signals.

When working with digital machines, the data are available only with some finite precision, and therefore, without loss of generality, the data can be considered to be integers with some upper bound. To compute convolution in this digital domain, operations in the complex number field of the continuous domain can be imitated in a finite field, or more generally, in a finite ring of integers under additions and multiplications modulo some integer M, with an integer α of order N, replacing $e^{-j(2\pi)/N}$ in the Discrete Fourier Transform (DFT).

In this ring, when two integer sequences x(n) and h(n) are convolved, the output integer sequence y(n) is congruent

to the conventional convolution of x(n) and h(n), modulo M. In the ring of integers modulo M, conventional integers can be, unambiguously, represented if their absolute value is less than M/2. If the input integer sequence x(n) and the filter sequence h(n) are so scaled that $|y(n)|$ never exceeds M/2, one would get the same results by implementing convolution in the ring of integers mod M as that obtained with normal arithmetic.

By special choices of the length N, the mod M, and the value α, it is possible to have transforms that need only word shifts and additions but no multiplications, that have an FFT type algorithm, that do not require storage of complex values of α and that have no round-off errors.

It has been shown that Mersenne transforms with $M = p = 2^q-1$, q a prime, and $\alpha = -2$, have the transform length equal to N = 2q and therefore do not have an FFT type fast computational algorithm.

The best known number theoretic transform is the Fermat Number Transform (FNT), where $M = 2^{2^t} + 1$, t a positive integer. For FNT's with a prime or composite modulus it was verified that $\alpha = 2$ or a power of 2 is possible, for sequences up to $N = 2^{t+1}$. This is a very desirable situation since N is highly composite allowing an FFT type algorithm and all multiplications by powers of α are simple word shifts. If $\alpha = \sqrt{2}$ is used then sequences of length $N = 2^{t+2}$ are possible, thus increasing the maximum sequence length permissible.

Assembler programs on the IBM 370/155 computer, written by Agarwal, showed that for cyclic convolutions length up to 256, the FNT implementations of convolution have a factor of 3 to 5 speed advantage over the FFT implementations. The reasons for the speed up are:

1 - The Fermat number transform requires no multiplications and, therefore, the implementation of convolution requires only N multiplications for an N point convolution. The number of additions and subtractions (together) for a convolution is $2N \log_2 N$ and there are $N \log_2 N$ required "multiplications" by a power of two.

2 - Only real operations are required. This buys about two to one savings over the FFT requirements.

3 - The Fermat number transform is able to compute an exact convolution thus allowing a program to avoid the need for either floating point arithmetic or overflow checks or other precautions.

The computation required to multiply the two transforms is about the same for both implementations. To convolve long sequences using two dimensional FNT, the computational effort increases by, at most, a factor of 2. Still the FNT implementation of convolution is much faster as compared to the FFT implementation.

Fermat number transforms have some additional advantages over the FFT. First, the FFT implementation requires storing

all powers of α requiring a significant amount of storage which may be an important factor for a small minicomputer or a special purpose hardware implementation.

Second, fixed-point FFT implementation introduces a significant amount of round-off noise at the output 6-8 bits depending on the data [38]. This degrades the signal-to-noise ratio during the filtering operations. The FNT is error free, the only source of error is input A/D quantization.

In the realm of radar signal processing the potential for higher throughput is worth exploring. For this reason McClellan designed and constructed a small prototype FNT-convolver. An important element in the design of the FNT-convolver was a new coding scheme for the data, although simpler codes are possible. The experience derived from designing and building this hardware serves as the basis for estimates of the site of large pipeline FNT convolvers, for use in radar matched filtering applications. The result of the hardware comparison of the FNT versus a pipeline FFT, indicates that the anticipated savings of the FNT can be realized for small systems (e.g., length 64 convolution), when the signal to be filtered is real. However, in larger systems where one must use two-dimensional convolution to implement one dimensional convolution, the savings in multiplier hardware are offset by increased transform size and the corresponding increase in memory size and reference

spectrum multiplier hardware. In this case, when the signal to be filtered is real, the FNT still offers a potential savings in the amount of hardware versus the FFT. When the signal is complex valued the amount of FNT hardware approximately doubles and is much greater than the pipeline FFT. In the implementation of two-dimensional convolution there is no penalty due to increased memory size and the savings in multiplier hardware will translate into savings for the overall convolver system.

It was shown that the main drawbacks of FNT's is a rigid relationship between word length and the obtainable transform length and a limited choice of possible word lengths. This last point is especially significant, since FNT's are restricted to word sizes equal to $q = 2^t$, t an integer. As q increases very rapidly with t, the choice of possible word lengths is very limited, and most practical digital filtering applications, when implemented with FNT, are constrained to word lengths of 16, 32 or 64 bits. If the dynamic range required for convolution does not correspond to $q = 2^t$ choosing the next higher value of q may result in a significant waste of computing power.

Various solutions to these problems involving either two-dimensional techniques, the use of "the Chinese Remainder Theroem", or other NTT's has been discussed.

Along these lines, it was shown that transforms over the Galois Field $GF(p^2)$, can be found which do not introduce round-off errors and which can be used to compute

information-lossless convolutions of sequences of complex numbers, by a FFT type algorithm. A disadvantage of this transform is that multiplications by powers of the primitive element (α) is not as simple as in Mersenne or FNT's.

It was verified that a Fourier-like transform in GF(p) where p is a prime of the form $A_n = 3 \times 2^n + 1$, n a positive integer, is possible. This transform increases the variety of methods and the digital word lengths that can be used for computing the convolution of integers beyond the above mentioned Fermat or Mersenne Number Transforms. Also, a special NTT that can be computed using a high-radix fast Fourier type algorithm, defined on arithmetic modulo primes of the form $(2^n-1)2^n + 1$, was discussed.

Complex Mersenne transforms that can be computed without multiplications were presented. These transforms are very promising for computing convolutions because they can be partly computed with the FFT type algorithms and some of the operations can be performed on words of reduced length. Complex Mersenne transforms also have the advantage of permitting operations on transform lengths and convolution lengths up to four times longer than is possible with conventional Mersenne transforms.

These results have been extended to cover the case of transforms oerating in a ring modulo a pseudo Mersenne number or submultiple of such a number. It was verified that some of these transforms have a highly composite transform

length and therefore can be computed with an efficient FFT-type algorithm. In the same way pseudo FNT's defined in a ring which is a submultiple of a Fermat number and can be considered as a generalization of FNT's. allow a much wider choice of possible word lengths, and therefore are well adapted for evaluating convolutions. As an extension the case of complex transforms was considered.

Finally, complex pseudo FNT's were presented, that allow a length double that of FNT's, and part of the calculations to be performed on words of reduced length. Some of these transforms have a highly composite number of terms and are therefore well suited for computing complex convolutions with an efficient FFT-type algorithm.

Recently [32] a number of three bit primes have been discovered which make possible very efficient fast number transforms approaching that of the FNT, but permitting much larger transform lengths suitable for convolving the large arrays met in picture processing and electron microscopy in particular. If a method of implementing arithmetic modulo three bit primes comparable to that already developed for implementing arithmetic modulo a Fermat number, could be found, very efficient fast convolution would become possible for a very large range of array sizes.

Rader and Brenner [41] have introduced an alternative form for the FFT. This new form has the advantage that none of the multipliers is complex, but in the usual complex field, most are pure imaginary. It has the disadvantage

that one must divide by very small numbers and therefore aggravate any quantization noise problems. This new algorithm may be applied to the complex NTT without this disadvantage.

Winograd [42] has shown how to perform a short-length D.F.T. of length $N = p$ or p^r (where p is a prime) in a very efficient way. Winograd's algorithm uses the fact that $\alpha^N = 1$ and requires that all operations be performed in a field. Hence, providing one chooses the modulus M to be a prime number, one may perform the NTT by using Winograd's algorithm.

A final remark is in order. Whether or not an engineering method is useful becomes clear only when that method is evaluated by the wide community of potential users, who consider it in relation to their needs. Since so many engineers and programmers are not familiar with number theory it is an open question whether NTT's algorithms will ultimately prove to be of great or small importance.

APPENDIX A

TWO DIMENSIONAL CONVOLUTION FOR CONVOLVING LONG SEQUENCES

Arithmetic mod F_t (a Fermat number) can be implemented using $b = 2^t$ bit representation of integers. In Section IV, it has been shown that the maximum length of sequences which can be cycled convolved using the FNT with $\alpha = 2$ is $N = 2b$ and therefore the length of sequences which can be convolved is proportional to the word length in bits. Thus for long sequences the word length requirement may be excessive.

Rader [3] suggested using a two dimensional convolution scheme to convolve long one-dimensional sequences and Agarwal and Burrus [17,18] presented such a two dimensional convolution scheme. Using this scheme, cyclic convolution of length $N = LP$ is implemented as a two dimensional cyclic convolution of length $2L \times P$.

This two-dimensional cyclic convolution can be implemented using a two dimensional FNT. Using this two dimensional scheme, the word length required is proportional to the square root of the length of the sequences to be convolved which would give for a maximum sequence length $8b^2$ rather than $4b$. I.e., for a computer word length $b = 64$

$$N \text{ for } \alpha = \sqrt{2} \text{ would be } \underline{32768}!$$

In the following pages an example illustrative of the two dimensional convolution using arithmetic modulo a Fermat number and FNT's is worked out.

Let $x(n)$ and $h(n)$, $n = 0, 1, \ldots N-1$, be two sequences which need to be cyclically convolved. Let $N = LP$.

We construct two $(2I \times P)$ two-dimensional sequences $\tilde{h}(i,j)$ and $\tilde{x}(K,\ell)$ from $x(n)$ and $h(n)$ respectively as shown below.

$$\tilde{h}(i,j) = h(jL + i - L) \qquad (1)$$
$$i = 0,1, \ldots, 2L-1$$
$$j = 0,1, \ldots, p-1.$$

And

$$\tilde{x}(K,\ell) = \begin{cases} x(\ell L+K) & K = 0,1, \ldots, L-1 \\ 0 & K = L, L+1, \ldots, 2L-1 \end{cases} \qquad (2)$$
$$\ell = 0,1, \ldots, p-1$$

Example:

Two dimensional convolution using FNT's.

Let

$x(n) = (2,-2,1,0)$ and $h(n) = (1,2,0,0)$

Using Fermat number transforms modulo $F_t = 17$,

$x(n) = (2,15,1,0)$ and $h(n) = (1,2,0,0)$

Note that the second element is changed from -2 to 15.

Notice that $N = 4 = L \times P = 2 \times 2$.

One constructs two $(2L \times p)$ two dimensional sequences $\tilde{h}(i,j)$ and $\tilde{x}(K,\ell)$ from $x(n)$ and $h(n)$, according to (1) and (2).

It turns out

$$\tilde{h}(i,j) = \begin{vmatrix} 0 & 1 \\ 0 & 2 \\ 1 & 0 \\ 2 & 0 \end{vmatrix} \qquad \tilde{x}(K,\ell) = \begin{vmatrix} 2 & 1 \\ 15 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$$

Now taking two dimensional transforms of $\tilde{h}$ and $\tilde{x}$ yields $\tilde{H}$ and $\tilde{X}$.

$$\tilde{H}(m,n) = \sum_{j=0}^{p-1} \sum_{i=0}^{2L-1} \tilde{h}(i,j) \alpha_{2L}^{im} \alpha_p^{jn} \qquad (3)$$

$m = 0,1, \ldots, 2L-1$ $\quad x_{2L}$ - is an element of order 2L

$n = 0,1, \ldots, p-1$ $\quad \alpha_p$ - is an element of order p

and a similar expression for $\tilde{x}(m,n)$.

Remembering that one is using $F_t = 17$ as the modulus

$$\alpha_{2L} = \alpha_4 = 13 \quad \text{and} \quad \alpha_p = \alpha_2 = 16$$

This can be found by noting that 3 and 6 primitive roots of the ring in consideration will generate all the positive integers in the ring (2.16).

Also, by (2.20), if α is a root of order N then

α^K is of order N/K if K|N (K divides N)

α^K is of order N if N and K are relatively prime.

Thus, with these considerations in mind one finds

$$\alpha_{2L} = \alpha_4 = 3^{16/4} = 3^4 = 13 \pmod{17}$$

$$\alpha = 13 \text{ order } 4$$

$$\alpha_p = \alpha_2 = 3^{16/2} = 3^8 = 16 \pmod{17}$$

$$\alpha = 16 \text{ order } 2$$

Now one can return to the calculation of the two dimensional transforms. The first term will be

$$\begin{aligned}\tilde{H}(0,0) = {} & \tilde{h}(0,0)\ 13^{(0)(0)}\ 16^{(0)(0)} + \tilde{h}(0,1)\ 13^{(0)(0)}\ 16^{(1)(0)} \\ & + \tilde{h}(1,0)\ 13^{(1)(0)}\ 16^{(0)(0)} + \tilde{h}(1,1)\ 13^{(1)(0)}\ 16^{(1)(0)} \\ & + \tilde{h}(2,0)\ 13^{(2)(0)}\ 16^{(0)(0)} + \tilde{h}(2,1)\ 13^{(2)(0)}\ 16^{(1)(0)} \\ & + \tilde{h}(3,0)\ 13^{(3)(0)}\ 16^{(0)(0)} + \tilde{h}(3,1)\ 13^{(3)(0)}\ 16^{(1)(0)}.\end{aligned}$$

i.e.,

$$\tilde{H}(0,0) = \tilde{h}(0,0) + \tilde{h}(0,1) + = 0 + 1 + = \underline{6} \pmod{17}$$

$$+ \tilde{h}(1,0) + \tilde{h}(1,1) + \quad 0 + 2 +$$

$$+ \tilde{h}(2,0) + \tilde{h}(2,1) + \quad 1 + 0 +$$

$$+ \tilde{h}(3,0) + \tilde{h}(3,1) \quad 2 + 0$$

Now constructing a table that will help in the evaluation of M(m,n).

N =	0	1	2	3	4	5	6	7	8	9
13^N =	1	13	16	4	1	13	16	4	1	13

notice order 4.

N =	0	1	2	3	4	5	6	7	8	9
16^N =	1	16	1	16	1	16	1	16	1	16

notice order 2

Now proceding:

$$\tilde{H}(1,0) = h(2,0)\ 13^{(2)(1)}\ 16^{(0)(0)} + \tilde{h}(0,1)\ 13^{(0)(1)}\ 16^{(1)(0)}$$

$$+ h(3,0)\ 13^{(3)(1)}\ 16^{(0)(0)} + \tilde{h}(1,1)\ 13^{(1)(1)}\ 16^{(1)(0)}$$

since $h(0,0) = h(1,0) = h(2,1) = h(3,1) = 0$.

$$\tilde{H}(1,0) = 1\ 13^{(2)}\ 16^{(0)} + 1\ 13^{(0)}\ 16^{(0)} + \ = \ 16 + 1 = \underline{0} \bmod 17$$

$$2\ 13^{(3)}\ 16^{(0)} + 2\ 13^{1}\ 16^{(0)} \quad + \ 8 + 26$$

For

$$\tilde{H}(2,0) = h(2,0)\ 13^{(2)(2)}\ 16^{(0)(0)} + h(0,1)\ 13^{(0)(2)}\ 16^{(1)(0)}$$

$$+ \ h(3,0)\ 13^{(3)(2)}\ 16^{(0)(0)} + h(1,1)\ 13^{(2)(2)}\ 16^{(1)(0)}$$

$$= 1\ 13^{(4)}\ 16^{(0)} + 1\ 13^{(0)}\ 16^{(0)} \ = \ 1 + 1 + \ = \underline{15} \ (\text{mod } 17)$$

$$+ \ 2\ 13^{(6)}\ 16^{(0)} + 2\ 13^{(2)}\ 16^{(0)} \quad + 32 + 32$$

and,

$$\tilde{H}(3,0) = h(2,0)\ 13^{(2)(3)}\ 16^{(0)(0)} + h(0,1)\ 13^{(0)(3)}\ 16^{(1)(0)}$$

$$+ \ h(3,0)\ 13^{(2)(3)}\ 16^{(4)(0)} + h(1,1)\ 13^{(1)(3)}\ 16^{(1)(0)}$$

$$= 1\ 13^{(6)}\ 16^{(0)} + 1\ 13^{0}\ 16^{0} \ = \ 16 + 1 \ = \ \underline{0} \ (\text{mod } 17)$$

$$+ \ 2\ 13^{9}\ 16^{0} + 2\ 13^{3}\ 16^{0} \quad + \ 26 + 8$$

For the second column,

$$\tilde{H}(0,1) = h(2,0)\ 13^{(2)(0)}\ 16^{(0)(1)} + h(0,1)\ 13^{(0)(0)}\ 16^{(1)(1)}$$

$$+ h(3,0)\ 13^{(3)(0)}\ 16^{(0)(1)} + h(1,1)\ 13^{(1)(0)}\ 16^{(1)(1)}$$

$$= 1 + 16 = \underline{0} \pmod{17}$$

$$+ 2 + 32$$

$$\tilde{H}(1,1) = 1\ 13^{(2)(1)}\ 16^{(0)(1)} + (1)\ 13^{(0)(1)}\ 16^{(1)(1)}$$

$$+ 2\ 13^{(3)(1)}\ 16^{(0)(1)} + (2)\ 13^{(1)(1)}\ 16^{(1)(1)}$$

$$= 16 + 16 = \underline{14} \pmod{17}$$

$$+ 8 + 416$$

$$\tilde{H}(2,1) = (1)\ 13^{(2)(2)}\ 16^{(0)(1)} + 1\ 13^{(0)(2)}\ 16^{(1)(1)}$$

$$+ 2\ 13^{(3)(2)}\ 16^{(0)(1)} + 2\ 13^{(1)(2)}\ 16^{(1)(1)}$$

$$= 1 + 16 = \underline{0} \pmod{17}$$

$$+ 32 + 512$$

$$\tilde{H}(3,1) = 1\ 13^{(2)(3)}\ 16^{(0)(1)} + 1\ 13^{(0)(3)}\ 16^{(1)(1)} = 16 + 16 = 16 \pmod{17}$$

$$+ 2\ 13^{(3)(3)}\ 16^{(0)(1)} + 2\ 13^{(1)(3)}\ 16^{(1)(1)} + 26 + 128$$

i.e.,

$$\tilde{H}(m,n) = \begin{vmatrix} 6 & 0 \\ 0 & 14 \\ 15 & 0 \\ 0 & 16 \end{vmatrix}$$

two dimensional
Fermat transform

The two dimensional transform of

$$\hat{\hat{X}} = \begin{vmatrix} 2 & 1 \\ 15 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$$

will be given by

$$\tilde{X}(m,n) = \sum_{j=0}^{p-1} \sum_{i=0}^{2L-1} \tilde{x}(i,j)\ \alpha_{2i}^{im}\ \alpha_{p}^{jn}$$

So

$$\tilde{X}(0,0) = \tilde{x}(0,0)\ 13^{(0)(0)}\ 16^{(0)(0)} + \tilde{x}(0,1)\ 13^{(0)(0)}\ 16^{(1)(0)}$$

$$+ \tilde{x}(1,0)\ 13^{(1)(0)}\ 16^{(0)(0)}$$

$$= 2 + 1 + 15 = \underline{1} \pmod{17}$$

$$\tilde{x}(1,0) = x(0,0)\ 13^{(0)(0)}\ 16^{(0)(0)} + x(0,1)\ 13^{(0)(1)}\ 16^{(1)(0)}$$

$$+ x(1,0)\ 13^{(1)(1)}\ 16^{(0)(0)}$$

$$= 2 + 1 + 15(13) = \underline{11} \pmod{17}$$

$$\tilde{x}(2,0) = x(0,0)\ 13^{(0)(2)}\ 16^{()0(0)} + x(0,1)\ 13^{(0)(2)}\ 16^{(1)(0)}$$

$$+ x(1,0)\ 13^{(1)(2)}\ 16^{(0)(0)}$$

$$= 2 + 1 + (15)(16) = \underline{6} \pmod{17}$$

$$\tilde{x}(3,0) = 2\ 13^{(0)(3)}\ 16^{(0)(0)} + (1)\ 13^{(0)(3)}\ 16^{(1)(0)}$$

$$+ 15\ 13^{(1)(3)}\ 16^{(0)(0)}$$

$$= 2 + 1 + (15)4 = \underline{12} \pmod{17}$$

$$\tilde{x}(0,1) = 2\ 13^{(0)(0)}\ 16^{(0)(1)} + (1)\ 13^{(0)(0)}\ 16^{(1)(1)}$$

$$+ 15\ 13^{(1)(0)}\ 16^{(0)(1)}$$

$$= 2 + 16 + 15\ \underline{16} \pmod{17}$$

$$\tilde{x}(1,1) = (2)\ 13^{(0)(1)}\ 16^{(0)(1)} + (1)\ 13^{(0)(1)}\ 16^{(1)(1)}$$

$$+ (15)\ 13^{(1)(1)}\ 16^{(0)(1)}$$

$$= 2 + 16 + 8 = \underline{9} \pmod{17}$$

$$\tilde{x}(2,1) = (2)\ 13^{(0)(2)}\ 16^{(0)(1)} + (1)\ 13^{(0)(2)}\ 16^{(1)(1)}$$

$$+ (15)\ 13^{(1)(2)}\ 16^{(0)(1)}$$

$$= 2 + 16 + (15)\ (16) = \underline{3} \pmod{17}$$

$$\tilde{x}(3,1) = (2)\ 13^{(0)(3)}\ 16^{(0)(1)} + (1)\ 13^{(0)(3)}\ 16^{(1)(1)}$$

$$+ (15)\ 13^{(1)(3)}\ 16^{(0)(1)}$$

$$= 2 + 16 + (15)(4) = \underline{10} \pmod{17}$$

So

$$\tilde{X}(m,n) = \begin{vmatrix} 1 & 16 \\ 11 & 9 \\ 5 & 3 \\ 12 & 10 \end{vmatrix}$$

two dimensional
Fermat transform

Let $\tilde{y}(i,j)$ be two-dimensional cyclic convolution of $\tilde{x}$ and $\tilde{h}$ sequences, then it can be proved that two dimensional transform of $\tilde{y}$ is the product of $\tilde{H}(m,n)$ and $\tilde{X}(m,n)$ [17], defined by

$$\tilde{y}(i,j) = \frac{1}{2PL} \sum_{n=0}^{p-1} \sum_{m=0}^{2L-1} \tilde{H}(m,n)\tilde{X}(m,n)\alpha_{2i}^{-mi}\ \alpha_{p}^{-jn}$$

$$i = 0,1, \ldots, 2L-1$$
$$j = 0,1, \ldots, p-1$$

Coming back to the example:

$$\tilde{y}(0,0) = \tilde{H}(0,0)\tilde{X}(0,0)13^{-(0)(0)}16^{-(0)(0)} + \tilde{H}(0,1)X(0,1)13^{-(0)(0)}16^{-(0)(0)}$$

$$+ \tilde{H}(1,0)\tilde{X}(1,0)13^{-(0)(1)}16^{-(0)(0)} + \tilde{H}(1,1)X(1,1)13^{-(0)(1)}16^{-(0)(1)}$$

$$+ \tilde{H}(2,0)\tilde{X}(2,0)13^{-(0)(2)}16^{-(0)(0)} + \tilde{H}(2,1)X(2,1)13^{-(0)(1)}16^{-(0)(1)}$$

$$+ \tilde{H}(3,0)\tilde{X}(3,0)13^{-(0)(3)}16^{-(0)(0)} + \tilde{H}(3,1)X(3,1)13^{-(0)(0)}16^{-(0)(1)}$$

i.e.,

$$\tilde{y}(0,0) = (6)(1)(1)(1) + (14)(4)(1)(1) + = \underline{10} \pmod{17}$$

$$+ (15)(5)(1)(1) + (16)(10)(1)(1)$$

$$\tilde{y}(1,0) = (6)(1)13^{-(1)(0)}16^{-(0)(0)} + (14)(9)13^{-(1)(1)}16^{-(0)(1)}$$

$$+ (15)(5)13^{-(2)(1)}16^{-(0)(0)} + (16)(10)13^{-(1)(3)}16^{-(0)(1)}$$

$$= 6 + 126\ 13^{-1} + 75\ 13^{-2} + 160\ 13^{-3}$$

$$= \underline{16} \pmod{17}$$

Notice that in this last calculation use of the following table has been made:

$$13^{-1} = ? \qquad 13 \times 13^{-1} = 1 \pmod{17} = 1\ 13^{-1} = 4$$

So

$13^{-1} = 4$ also $16^{-1} = 16$

$13^{-2} = 4^2 = 16$ $16^{-2} = 1$ } order 2

$13^{-3} = 4^3 = 13$ $16^{-3} = 16$

$13^{-4} = 4^4 = 1$ $16^{-4} = 1$

$13^{-5} = 4^5 = 4$ order 4 $16^{-5} = 16$

$13^{-6} = 4^6 = 16$ $16^{-6} = 1$

$13^{-7} = 4^7 = 13$ $16^{-7} = 16$

$13^{-8} = 4^8 = 1$ $16^{-8} = 1$

$13^{-9} = 4^9 = 4$ $16^{-9} = 16$

Continuing with the example:

$$\tilde{y}(2,0) = (6)(1)\,13^{-(2)(0)}16^{-(0)(0)} + (14)(9)\,13^{-(2)(1)}16^{-(0)(1)}$$

$$+ (15)(5)\,13^{-(2)(2)}16^{-(0)(0)} + (16)(10)13^{-(2)(3)}16^{-(0)(1)}$$

$$= \underline{16} \pmod{17}$$

$$\tilde{y}(3,0) = (6)(1)\,13^{-(3)(0)}16^{-(0)(0)} + (14)(9)\,13^{-(3)(1)}16^{-(0)(1)}$$

$$+ (15)(5)\,13^{-(3)(2)}16^{-(0)(0)} + (16)(10)13^{-(3)(3)}16^{-(0)(1)}$$

$$= \underline{16} \pmod{17}$$

and for the second column,

$$\tilde{y}(0,1) = (6)\ (1)\ 13^{-(0)(0)}16^{-(1)(0)} + (14)(9)\ 13^{-(0)(1)}16^{-(1)(1)}$$

$$+ (15)(5)\ 13^{-(0)(2)}16^{-(1)(0)} + (16)(10)13^{-(0)(3)}16^{-(1)(1)}$$

$$= \underline{16} \pmod{17}$$

$$\tilde{y}(1,1) = (6)\ (1)\ 13^{-(1)(0)}16^{-(1)(0)} + (14)(9)\ 13^{-(1)(1)}16^{-(1)(1)}$$

$$+ (15)(5)\ 13^{-(1)(2)}16^{-(1)(0)} + (16)(10)13^{-(1)(3)}16^{-(1)(1)}$$

$$= \underline{16} \pmod{17}$$

$$\tilde{y}(2,1) = (6)(1)\ 13^{-(2)(0)}16^{-(1)(0)} + (14)(9)\ 13^{-(2)(4)}16^{-(1)(1)}$$

$$+ (15)(5)13^{-(2)(2)}16^{-(1)(0)} + (16)(10)13^{-(2)(3)}16^{-(1)(1)}$$

$$= \underline{10} \pmod{17}$$

$$\tilde{y}(3,1) = (6)(1)\ 13^{-(3)(0)}16^{-(1)(0)} + (14)(9)\ 13^{-(3)(1)}16^{-(1)(1)}$$

$$+ (15)(5)13^{-(3)(2)}16^{-(1)(0)} + (16)(10)13^{-(3)(3)}16^{-(1)(1)}$$

$$= \underline{16} \pmod{17}$$

Finally,

$$\tilde{y}(i,j) = \frac{1}{2PL} \begin{vmatrix} 10 & 16 \\ 16 & 16 \\ 16 & 10 \\ 16 & 16 \end{vmatrix}$$

and since

$$\frac{1}{2PL} = \frac{1}{2N} = (2N)^{-1} = (8^{-1}) = 15 \pmod{17}$$

we have

$$\tilde{y}(i,j) = 15 \begin{vmatrix} 10 & 16 \\ 16 & 16 \\ 16 & 10 \\ 16 & 16 \end{vmatrix} = \begin{vmatrix} 14 & 2 \\ 2 & 2 \\ 2 & 14 \\ 2 & 2 \end{vmatrix} \pmod{17}$$

And, since it can be proved that the relationship between two dimensional convolution and one dimensional convolution is given by [18]:

$$\tilde{y}(i+L,j) = y(jL+i)$$

where

$i = 0,1, \ldots, L-1$

$j = 0,1, \ldots, p-1.$

In this example:

	$\tilde{y}(i+L,j)$	$y(jL+i)$
① $i = 0, j = 0$	$\tilde{y}(0+2,0) = \tilde{y}(2,0)$	$y(0\cdot 2+0) = y(0)$
	i.e. $\tilde{y}(2,0) = y(0) = \underline{2}$	
② $i = 1, j = 0$	$\tilde{y}(1+2,0) = \tilde{y}(3,0)$	$y(0\cdot 2+1) = y(1)$
	i.e. $\tilde{y}(3,0) = y(1) = \underline{2}$	
③ $i = 0, j = 1$	$\tilde{y}(0+2,1) = \tilde{y}(2,1)$	$y(1\cdot 2+0) = y(2)$
	i.e. $\tilde{y}(2,1) = Y(2) = \underline{14}$	
④ $i = 1, j = 1$	$\tilde{y}(1+2,1) = \tilde{y}(3,1)$	$y(1\cdot 2+1) = y(3)$
	i.e. $\tilde{y}(3,1) = y(3) = \underline{2}$	

Note that for the original sequences

$$x(n) = (0,1,15,2) \quad \text{and} \quad h(n) = (1,2,0,0)$$

$$y(n) = x(n) * h(n) = 2,2,14,2.$$

Exactly the same result!!

In taking two dimensional transforms, p and 2L are restricted to be a power of $\underline{2}$ and also $p \leq 4b$ and $2L \leq 4b$ ($b = 2^t$ bit representation of integers in arithmetic modulo F_t Fermat number) in the example:

Since $F_t = 2^b + 1 = 17 \quad b = 4 \quad t = 2,$

also $p = 2, \quad L = 2.$

Notice that $N = PL \leq 8b^2$.

Thus the length of the sequences that can be convolved using two dimensional convolution scheme is proportional to the square of the number of bits used in the word length.

The order in which two dimensional transforms or inverse transforms is taken is reversible. But there is some computational advantage in taking transform first along the direction 2 (length p) and then along the direction 1 (length 2L), because half the $\tilde{x}$ sequences along direction 2 (p) are zero and half the $\tilde{n}$ sequences along direction 2 are cyclic shifts by one position of the other half $\tilde{n}$ sequences [18].

Also while taking the inverse transform, computationally it is advantageous to first take the inverse transform along direction 1, then along direction 2 [18]. Because we need only half the $\tilde{y}$ sequences along direction 2, therefore after taking inverse transforms along direction 1, we can throw away half the terms.

APPENDIX B

BASIC PROPERTIES OF QUADRATIC RESIDUES

Let

$$x^2 \equiv a \pmod{p} \tag{B.1}$$

be a congruence, where p is any odd prime and a is any integer. If $a \equiv 0 \pmod{p}$, then the only solution to (B.1) is $x \equiv 0 \pmod{p}$. Therefore one assumes $p \nmid a$. For some values of a, (B.1) will have a solution, whereas for some other values of a, (B.1) will have no solution.

Definition B.1: Let p be a prime, and let a be any integer such that $p \nmid a$. One says that a is a quadratic residue modulo p provided that

$$x^2 \equiv a \pmod{p} \tag{B.2}$$

has a solution. Otherwise, one says that a is a quadratic nonresidue modulo p.

Suppose that p is given and consider the problem of determining all quadratic residues modulo p. If a is a quadratic residue modulo p, then $p \nmid a$ and $a \equiv x^2 \pmod{p}$ for some x. However, since any integer is congruent to one of $0, 1, \ldots, p-1 \pmod{p}$, one sees that a must be congruent to one of

$$1^2, 2^2, \ldots, (p-1)^2 \pmod{p} \qquad \text{(B.3)}$$

If p is not too large, then this procedure can actually be used for computation.

Example: Let $p = 13$

Then a is a quadratic residue modulo 13 if and only if a is congruent to one of

$$1^2, 2^2, \ldots, 12^2 \pmod{13} ;$$

that is a is a quadratic residue modulo 13 if and only if $a \equiv 1,4,9,3,12,10,10,12,3,9,4,1 \pmod{13}$.

Thus the quadratic residues of 13 are

$$1, 3, 4, 9, 10, 12.$$

Hence the quadratic nonresidues modulo 13 are

$$2, 5, 6, 7, 8, 11.$$

Notice that the initial list of quadratic residues obtained is symmetric, with each element of the list appearing exactly twice. This is a general phenomenon.

Indeed, one has

$$p-x \equiv -x \pmod{p} \qquad \text{(B.4)}$$

so that

$$(p-x)^2 \equiv (-x)^2 \bmod p \qquad (B.5)$$

and thus

$$(p-x)^2 \equiv x^2 \pmod p \qquad (B.6)$$

therefore, if a is a quadratic residue modulo p, then a is congruent to one of

$$1^2, 2^2, \ldots, (\frac{p-1}{2})^2 \bmod p \qquad (B.7)$$

No two integers of (B.7) are congruent modulo p. Hence, among the integers

$$1, 2, \ldots, p-1$$

precisely (p-1)/2 are quadratic residues modulo p and precisely (p-1)/2 are quadratic nonresidues modulo p. This can be verified in the above example. One finds the following notation very convenient:

Definition B.2:

Let p be an odd prime and a an integer such that $p \nmid a$. The Legendre symbol is defined as follows:

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a \text{ is a quadratic residue modulo } p \\ -1 & \text{if } a \text{ is a quadratic nonresidue modulo } p \end{cases} \qquad \text{(B.8)}$$

The Legendre symbol $\left(\frac{a}{p}\right)$ should not be confused with the fraction a/p.

Example: Let p = 3

$$\left(\frac{2}{13}\right) = -1\,, \quad \left(\frac{3}{13}\right) = 1\,, \quad \left(\frac{4}{13}\right) = 1\,,$$

$$\left(\frac{5}{13}\right) = -1$$

See above example, where it is listed the quadratic residues modulo 13, and quadratic nonresidues mod 13. Moreover, since

$$18 \equiv 5 \pmod{13}$$

and 5 is a quadratic nonresidue modulo 13. So is 18, and thus

$$\left(\frac{18}{13}\right) = -1$$

Properties of Legendre symbol.

Let p be an odd prime and let a and b be integers such that $p \nmid a$ and $p \nmid b$. Then the following results hold [20]:

(i) $\left(\frac{a^2}{p}\right) = 1$

(ii) $\left(\frac{1}{p}\right) = 1$ (B.9)

(iii) If $a \equiv b \bmod p$ then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$

Proof:

(i) The congruence $x^2 \equiv a^2 \bmod p$ has as a solution $x = a$.

(ii) Set $a = 1$ in result (i).

(iii) If $a \equiv b \pmod p$, then the solutions $x^2 \equiv a \pmod p$ are the same as the solutions of $x^2 \equiv b \pmod p$. Therefore, the first congruence has solutions if and only if the second does. Thus,

$$\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$$

The properties of the Legendre symbol given above are very elementary. However a property of the symbol which is by no means obvious is the following result:

Theorem 8.3: (Euler's Criterion):

Let p be an odd prime and let a be an integer such that $p \nmid a$. Then

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p} \qquad \text{(B.10)}$$

Proof:

By Fermat's theorem (2.14), one has

$$(a^{(p-1)/2})^2 = a^{p-1} \equiv 1 \pmod{p}$$

Thus, if

$$h = a^{(p-1)/2}$$

then

$$h^2 \equiv 1 \bmod p$$

and so

$$p \mid (h-1)(h+1).$$

Therefore

$$p \mid h-1,$$

or

$$p \mid h+1,$$

and hence

$$h = a^{(p-1)/2} \equiv \pm 1 \pmod{p}.$$

Now if p is odd, so $(\frac{a}{p}) = +1$ if and only if $a^{(p-1)/2} \equiv 1 \pmod{p}$. Consequently, $(\frac{a}{p}) = \pm 1$ if and only if

$$a^{(p-1)/2} \equiv \pm 1 \pmod{p}$$

respectively.

Corollary B.4:

Let p be an odd prime, and let a and b be integers such that $p \nmid a$, $p \nmid b$. Then

$$(\frac{ab}{p}) = (\frac{a}{p})(\frac{b}{p}) \qquad (B.11)$$

Proof:

By Euler's Criterion

$$(\frac{ab}{p}) = (ab)^{(p-1)/2} = a^{(p-1)/2} b^{(p-1)/2} \equiv (\frac{a}{p})(\frac{b}{p}) \pmod{p}.$$

It is an immediate consequence of Corollary (B.4) that

(i) the product of two quadratic residue modulo p is a quadratic residue modulo p.

(ii) the product of two quadratic nonresidue modulo p is a quadratic residue modulo p,

and

(iii) the product of a quadratic residue and a quadratic nonresidue is a quadratic nonresidue.

Example: Let $p = 13$

By previous calculations

3 and 12 are quadratic residues mod 13

and

$3 \cdot 12 = 36 = 6^2 \pmod{13}$ is a quadratic residue.

However

2 and 5 are quadratic nonresidues mod 13

and

$2 \cdot 5 = 10 = 6^2 \pmod{13}$ is a quadratic residue mod 13.

Finally

7 is a quadratic nonresidue mod 13

10 is a quadratic residue mod 13

and

$$7 \cdot 10 = 70 \equiv 5 \pmod{13} \text{ is a nonresidue.}$$

Another consequence of Euler's criterion is the following:

Corollary 8.5:

Let p be an odd prime. Then

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$$

In other words,

$$\left(\frac{-1}{p}\right) = \begin{cases} +1 & \text{if } p \equiv 1 \pmod 4 \\ -1 & \text{if } p \equiv 3 \pmod 4 \end{cases}$$

Proof:

By Euler's criterion,

$$\left(\frac{-1}{p}\right) \equiv (-1)^{(p-1)/2} \pmod p$$

Therefore, since $\left(\frac{-1}{p}\right) = \pm 1$ and since $p > 2$, we have the desired result.

Notice that

$$\left(\frac{-a^2}{p}\right) = \left(\frac{-1}{p}\right)\left(\frac{a^2}{p}\right) = \left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$$

since $(\frac{a^2}{p}) = 1$ by (B.9), therefore, $x^2 \equiv -a^2 \pmod{p}$ is solvable if and only if $p \equiv 1 \pmod{4}$.

Example:

Let

$$x^2 \equiv 19 \pmod{23}$$

Now

$$19 \equiv -4 \pmod{23}$$

So

$$(\frac{19}{23}) = (\frac{-4}{23}) = (\frac{-1}{23})\ (\frac{2}{23})^2 = -1$$

Since $23 \equiv 3 \pmod{4}$. Thus $x^2 \equiv 19 \pmod{23}$ is not solvable.

How does one go about computing $(\frac{a}{p})$ for $p \nmid a$?

Suppose that

$$a = \pm p_1^{a_1} \cdots p_t^{a_t}$$

where $p_1, \ldots, p_t$ are distinct primes. Since $p \nmid a$, one sees that $p \neq p_i$. Then by Corollary 4, one has

$$(\frac{a}{p}) = (\frac{\pm 1}{p})\ (\frac{p_1}{p})^{a_1} \cdots (\frac{p_t}{p})^{a_t}$$

Example:

Let $p = 5 \quad a = -24$

$$\left(\frac{-24}{5}\right) = \left(\frac{-1}{5}\right)\left(\frac{2}{5}\right)^3\left(\frac{3}{5}\right) = 1 \cdot (-1)^3(-1) = 1$$

LIST OF REFERENCES

1. A. Schonhage and V. Strassen, "Schnelle MultipliKation Grosser Zahlen," Computing (Arch. Elektron Rechmen), 7(1971), pp. 281-292.

2. D.E. Knuth, The Art of Computer Programming, Addison-Wesley, Reading, Mass., 1969.

3. C.M. Rader, "Discrete Convolution Via Mersenne Transforms," IEEE Trans. Computers, C-21 (1972), pp. 1269-1273.

4. R.C. Agarwal and C.S. Burrus, "Fast Convolution using Fermat Number Transforms with Applications to Digital Filtering," IEEE Trans. Acoustics, Speech, and Signal Processing, ASSP-22 (1974), pp. 87-97.

5. Gauss, Carl F., Disquisitiones Arithmeticae, Translated by Arthur A. Clarke, Yale University Press, New Haven, Connecticut, 1966.

6. Dickson, Leonard E., History of the Theory of Numbers, Carnegie Institution of Washington, Washington, D.C., 1919.

7. Szabo, Nicholas, and Tanaka, Richard, Residue Arithmetic and its Applications to Computer Technology, McGraw-Hill, New York, 1967.

8. Atkin, A.O.L., and Birch, B.J., "Computers in Number Theory," Proceedings of the Science Research Council Atlas Symposium No. 2, Held at Oxford 18-23 August 1969, Academic Press, New York, 1971.

9. Gardner, Harver L., "The Residue Number Systems," IRE Transactions on Electronic Computers, Vol. EC-8, pp. 140-147, June 1959.

10. Fraenkel, Aviezri, "The Use of Index Calculus and Mersenne Primes for the Design of High-speed Digital Computers," J. Acm. Vol. 8, No. 1, pp. 87-96, 1961.

11. Banerji, D.K. and Brozozowski, J.A., "On Translation Algorithms in Residue Number Systems," IEEE Trans. on Computers, pp. 1281-1285, Dec. 1972.

12. _____, "Sign Detection in Residue Number Systems," IEEE Trans. on Computers, Vol. I-10, No. 4, pp. 313-320, April 1969.

13. Agarwal, R.C., and Burrus, C.S., "Number Theoretic Transforms to Implement Fast Digital Convolution," Proceedings of the IEEE, Vol. 63, pp. 550-560, April 1975.

14. Pollard, J.M., "The Fast Fourier Transform in a Finite Field," Math. Comput., 1971, 25, pp. 365-374.

15. Good, I.J., "The Relation between Two Fast Fourier Transforms," IEEE Trans., 1971, C-20, pp. 310-317.

16. Melhuish, P., "Fermat Transform Implementation by a Minicomputer," Electron. Lett., 1975, 11, pp. 109-111.

17. Agarwal, R.C. and C.S. Burrus, "Fast Digital Convolutions using Fermat Transforms," in Southwest IEEE Conf. Rec. pp. 538-543, April 1973.

18. _____, "Fast One-Dimensional Digital Convolution by Multi-dimensional Techniques," IEEE Trans. Acoust., Speech, and Signal Processing, Vol. ASSP-22, pp. 1-10, Feb. 1974.

19. Golomb, S.W., I.S. Reed and T.K. Truong, "Integer Convolutions over the Finite Field GF $(3 \cdot 2^n+1)$," SIAM J. Appl. Math., 1977, 32, pp. 356-365.

20. Reed, Irving S, T.K. Truong, "The Use of Finite Fields to Compute Convolutions," IEEE Trans. on Information Theory, Vol. IT-21, No. 2, March 1975.

21. _____, "Convolutions over Residue Classes of Quadratic Integers," IEEE Trans. on Information Theory, Vol. IT-22, No. 4, July 1976.

22. Pollard, J.M., "Implementation of Number Theoretic Transforms," Electronics letters, Vol. 12, No. 15, 1976.

23. Liu, K.Y., I.S. Reed, T.K. Truong, "Fast Number Theoretic Transforms for Digital Filtering," Electronics Letters, Vol. 12, No. 24, Nov. 1976.

24. Nussbaumer, H.J., "Digital Filtering Using Complex Mersenne Transforms," IBM J. Res. Develop., 20, 498 (1976).

25. Pelep, A., and B. Liu, Digital Signal Processing, John Wiley & Sons, 1976.

26. McClellan, J.H., "Hardware Realization of a Fermat Number Transform," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-24, No. 3, June 1976.

27. Leibowitz, L.M., "A Simplified Binary Arithmetic for the Fermat Number Transform," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-24, No. 5, Oct. 1976.

28. Nussbaumer, H.J., "Linear Filtering Technique for Computing Mersenne and Fermat Number Transforms," IBM J. Res. Develop., July 1977.

29. Brule, J.D., "Fast Convolution with Finite Field Fast Transforms," IEEE Trans. on Acoustics, Speech and Signal Processing, April 1975.

30. Reed, J.S. and T.K. Truong, "Complex Integer Convolutions over a Direct Sum of Galois Fields," IEEE Trans. Inform. Theory, Vol. IT-21, pp. 657-661, Nov. 1975.

31. Agarwal, R.C. and C.S. Burrus, "Fast One-dimensional Digital Convolution by Multidimensional Techniques," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-22, pp. 1-10, Feb. 1974.

32. Derome, M.F.A., "Fast Convolution of Large One and Two-dimensional Arrays Using Number Theoretic Transforms (NTT) Based on Three Bit Primes," Optik 49 (1978), No. 4, 465-475.

33. Burrus- C.S., "Block Realization of Digital Filters," IEEE Trans. Audio Electroacoust., Vol. AU20, pp. 230-235, Oct. 1972.

34. Gold, B., and C.M. Rader, Digital Processing of Signals, McGraw Hill, 1969, Chapter 7.

35. Nicholson, P.J., "Algebraic Theory of Finite Fourier Transforms," J. Comput. Syst. Sci., Vol. 5, pp. 524-547, 1971.

36. Reed, I.S., and K.Y. Liu, "Fast Algorithm for Computing Complex Number-Theoretic Transforms," Electronics Letters, Vol. 13, No. 10, May 1977.

37. Nussba-mer, H.J., "Digital Filtering using Pseudo Fermat Number Transforms," IEEE Transactions on ASSP, Vol. ASSP-25, No. 1, Feb. 1977.

38. Rabiner, L.R. and B. Gold, Theory and Application of Digital Signal Processing, Englewood Cliffs, N.J.: Prentice Hall, 1974.

39. Oppenheim, A.V. and C. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform," Proc. IEEE, Vol. 60, No. 8, Aug 1972, pp. 957-976.

40. Blankenship, P.E., and E.M. Hofstetter, "Digital Pulse Compression via Fast Convolution," IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-23, pp. 189-201, April 1975.

41. Rader, C.M. and N.M. Brenner, "A New Principle for Fast Fourier Transformation," IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-24, pp. 264-266, April 1976.

42. Winograd, S., "On Computing the Discrete Fourier Transform," Proc. Nat. Acad. Sci., U.S.A. 1976, 73, pp. 1005-1006.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Professor S. R. Parker, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Professor T. F. Tao, Code 62Tv Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	LT. Antonio Catarino Rodrigues de Sousa Portuguese Navy RUA DOS BACALHOEIROS, 99-5-ESQ LISBOA-2, PORTUGAL	1